Approximation Algorithms for Minimizing Sum of Flow Time on a Single Machine with Release Dates

E. O. Oyetunji[1*] and A. E. Oluleye[2,b] and S.A. Akande[2,c]
[1]*Academic Quality Assurance Unit University for Development Studies, Ghana*
[2]*Department of Industrial and Production Engineering University of Ibadan, Nigeria*

Abstract: This paper considers the scheduling problem of minimizing the sum of flowtime on single machine with release dates. It is well known that the problem is NP-Hard, therefore in order to solve the problem, two approximation algorithms (KSA1 and KSA2) were proposed. KSA1 and KSA2 algorithms were compared with the MPSW (selected from the literature) and the Branch and Bound (BB) method. All the four solution methods were evaluated on a set of randomly generated problems. Twenty problem sizes ranging from 3 to 100 jobs and fifty problem instances under each problem size were generated. A total of 1000 (50x20) problem instances were solved. Experimental results based on effectiveness show that the MPSW algorithm performs closest to the BB method and outperformed both the KSA1 and KSA2 algorithms for all the problem sizes considered. On the other hand, based on efficiency, the KSA1 and KSA2 algorithms outperformed both MPSW and BB methods.

Keywords: *Single Machine Scheduling, Sum of Flowtime, Release Dates, Effectiveness, Efficiency, Approximation Algorithm*

1. INTRODUCTION

Scheduling can be defined as the allocation of a set of defined resources to a set of defined tasks subject to certain constraints, in order to satisfy a specific objective [9]. Scheduling has wide applications in computer systems, Hospital administration, Transportation management, Lecture and examination planning, Manufacturing e.t.c. Generally, scheduling problems involve jobs that must be scheduled on machines subject to certain constraints in order to optimize one or more objective function(s).

The methods for solving general scheduling problems can be classified into: exact and approximation methods. Exact method yield optimal solutions (e.g. total enumeration method, Hungarian method, Johnson's method for 2-machine sequencing, implicit enumeration method such as branch and bound or dynamic programming methods). The approximation method, on the other hand, involves the use of heuristic algorithms. Heuristic methods usually involve the use of an intuitive approach or rule of thumb. Heuristic methods are techniques for obtaining acceptable solutions to scheduling problems at reasonable computational costs. While they do not always guarantee optimal results, the techniques are relatively economical in terms of computational resources utilized. In view of the computational difficulty required by the exact methods, the benefit of lower computational costs has been an attraction to being utilized by researchers in order to solve scheduling problems [13]. This is the motivation for adopting approximation algorithms in this work.

There are several objectives (performance measures) in scheduling [12]. In systems involving queuing and networks, for example, the flow time of a job consists of both the waiting time in the queue and the job processing time on the machine so that minimizing flow time improves service quality [11]. The desire to improve service quality makes the minimization of the total flow time (or sum of flow time) an important scheduling criterion. The foregoing also motivated the selection of the sum of flow time as the criterion to be minimized. Scheduling jobs on a single machine is important because single machine environments are common and can usually be modeled as a special case of other environments. Because the assumption that all jobs are available at time zero does not always hold, we focus on problems for which jobs have distinct ready or release dates.
Therefore, the main aim of this work is to develop approximation algorithms that can be used to solve the scheduling problem of minimizing the sum of flow time on a single machine with release dates.

Three variants of the problem have been explored by researchers. These are:

(1) problems in which the release dates are zeros (i.e. $1|\ |\sum_{i=1}^{n} F_i$),

[1] E.O. Oyetunji holds a PhD degree in Industrial Engineering, and is currently a Senior Lecturer in the Department of Computer Science and Director, Academic Quality Assurance Unit, University for Development Studies, Ghana.
[2] A.E. Oluleye holds a PhD degree in Industrial Engineering, and is currently a Professor in the Department of Industrial and Production Engineering and Dean, Faculty of Technology, University of Ibadan, Nigeria.
[*] Corresponding author.

(2) problems in which pre-emption is allowed (i.e. $1|\ pmtn,\ r_i\ |\sum_{i=1}^{n} F_i$), and

(3) the general non preemptive problems in which the release dates are different and distinct (i.e. $1|r_i\ |\sum_{i=1}^{n} F_i$).

Smith [15] showed that the first variant of the problem can be solved optimally using the Shortest Processing Time (SPT) rule. The second variant of the problem (the preemptive version) can be solved optimally in polynomial time by using the Shortest Remaining Processing Time (SRPT) rule [2]. The solution of the preemptive version of the problem provides a lower bound for the third variant (the general non-preemptive version) of the problem [1]. The general non-preemptive version of the problem is known to NP-Hard [10]. We are unaware of any exact solution method for this third variant of the problem. A number of authors have developed branch-and-bound algorithms for the general non-preemptive version of the problem [3]-[6].

Guo et al. [8] noted that an optimal solution for $1|r_i\ |\sum_{i=1}^{n} C_i$ is also an optimal solution for $1|r_i\ |\sum_{i=1}^{n} F_i$. In view of this, Guo et al. [8] modified the PSW algorithm (proposed for $1|r_i\ |\sum_{i=1}^{n} C_i$ by Phillips et al., [14]) to solve the $1|r_i\ |\sum_{i=1}^{n} F_i$ problem. The algorithm is called MPSW.

2. PROBLEM DEFINITION

Given the general one-machine scheduling problem where a set *J* of *n* jobs has to be sequenced on a machine in order to minimize the sum of flow time (also called total flow time). We assumed that only one job can be processed at a time and that the arrival time of every job J_i at the machine is known, distinct and denoted by r_i (release date). Also, each job J_i requires p_i time units on the machine (processing time). The time the processing of job J_i starts on the machine (start time) is designated as s_i with the property:

$$s_i \geq r_i \qquad (1)$$

the completion time of each job (C_i) is defined as:

$$C_i = s_i + p_i . \qquad (2)$$

The flow time of each job is defined as:

$$F_i = C_i - r_i \qquad (3)$$

The sum of the flow time (also called total flow time) is defined as

$$F_{tot} = F_1 + F_2 + ... + F_n = \sum_{i=1}^{n} F_i \qquad (4)$$

Using the notations of Graham et al. [7], the problem being explored is represented as

$1|\ r_i\ |(F_{tot})$ or $1|r_i\ |\sum_{i=1}^{n} F_i$

It is assumed that pre-emption is not allowed and that the problem is static and deterministic i.e. number of jobs, their processing times, and ready times and due dates are all known and fixed.

3. MATERIALS AND METHODS

The methods adopted in this study are now described:

3.1. Solution Methods

In order to solve the scheduling problem of minimizing the sum of flow time on a single machine with release dates, two approximation algorithms (labeled KSA1 and KSA2) are proposed. In order to compare the performances of the proposed algorithms, the MPSW algorithm of Guo et al. [8] was selected while a branch and bound (BB) procedure was also implemented. All the four solution methods are now described.

3.1.1. Branch and Bound (BB) Method

The branch and bound (BB) procedure is an implicit enumeration scheme where certain schedules or classes of schedules are discarded by showing that the values of the objective function obtained with schedules from this class are worse than a provable lower bound. The BB procedure gives optimal results. In order to compare the performances of the proposed approximation algorithms, the branch and bound method was implemented. The Shortest Processing Time (SPT) rule was used to obtain the lower bound at each node. The SPT is optimal for the relaxed ($1 \mid \mid \sum_{i=1}^{n} F_i$) problem [9]. The node that gave the best lower bound value (total flow time) determines the branch to explore. At the terminal node, when all the jobs must have been assigned, the node that gave the best solution was noted and became the solution to the considered problem.

3.1.2 MPSW ALGORITHM

The MPSW algorithm was proposed by Guo et al. [9] for the scheduling problem of minimizing total flow time criterion on a single machine with release dates (i.e. $1 \mid r_i \mid \sum_{i=1}^{n} F_i$ problem). The MPSW algorithm produces non-preemptive schedules from preemptive ones as follows:

Step 1: Form a preemptive schedule using the Shortest Remaining Processing Time (SRPT) rule.
SRPT always picks jobs with the shortest remaining processing times among those already released at the current time and processes these first. Each job i has a (preemptive) completion time C_i^P.

Step 2: Form an ordered list L of jobs based on their preemptive completion time C_i^P using a simple sort.

A non-preemptive schedule is then obtained if we continue to assign the first job in L to the machine when it is freed and delete it from L.

3.1.3 KSA1 Algorithm

The KSA1 algorithm schedules the job that has the least value of the sum of the processing time and release date in the first position (breaking ties by selecting the job with the job that has the lowest release date among the tied jobs). The remaining jobs are scheduled in the increasing order of their waiting times. Waiting times of the remaining jobs are computed as the absolute value of the time difference between the release date of each job and the completion time of the first scheduled job. The steps of KSA1 algorithm are now described:

KSA1 Algorithm Steps

STEP 1: Initialization
Job_Set_A = [J_1, J_2, J_3, …….J_n], set of given jobs
Job_Set_B = [0], set of schedules job
Job_Set_C = [J_1', J_2', J_3', …….J_n'], set of unscheduled jobs, $J_j' = J_j$
Job_Set_D = [0]
n=number of jobs

STEP 2: Compute the index= p_i+r_i for each of the jobs in Job_Set_A, i = 1, .., n

STEP 3: Arrange the jobs in Job_Set_A in the order of increasing index computed in Step 2 and put the jobs in Job_Set_D. If there is a tie (i.e. two or more jobs have the same index value), select first the job that has the lowest release date among the tied jobs. If there are still ties, break ties arbitrarily.

STEP 4: Select the job in the first position from Job_Set_D, add this job to Job_Set_B and remove same job from Job_Set_C.

STEP 5: Compute the Completion time of the job scheduled in step 4 (C_1)

STEP 6: Compute $\Delta W_j = |R_j - C_i|$ for all the remaining jobs in Job_Set_D. Where R_j is the release date of each of the remaining jobs in Job_Set_D and C_i is the completion time of jobs in Job_Set_D , j= 2, 3, …, n-1, i= 1, 2, 3, …, n.

STEP 7 : Re-arrange the remaining jobs in Job_Set_D in the order of their increasing ΔW computed in Step 6

STEP 8 : Append Job_Set_D to Job_Set_B

STEP 9 : Stop

3.1.4 KSA2 Algorithm

The KSA2 algorithm is similar to the KSA1 algorithm. The main differences between the KSA2 and KSA1 algorithms are as follows: after obtaining the initial schedule, while the KSA1 break ties by selecting the job with the least release date, the KSA2 algorithm in addition proceeds to break ties by selecting the job with the least processing time. Also, the KSA2 algorithm computes the raw waiting time of jobs instead of the absolute values of the waiting time.
The steps of KSA2 algorithm are now described:

KSA2 Algorithm Steps

STEP 1 : Initialization
Job_Set_A = [J_1, J_2, J_3, …….J_n], set of given jobs
Job_Set_B = [0], set of schedules job
Job_Set_C = [J_1', J_2', J_3', …….J_n'], set of unscheduled jobs, $J_j' = J_j$
Job_Set_D = [0]
n=number of jobs
STEP 2: Compute the index = p_i+r_i for each
of the jobs in Job_Set_A, i= 1, 2 ……, n.
STEP 3: Arrange the jobs in the order of
increasing index computed in Step 2 and put the jobs in Job_Set_D. If there is a tie (i.e. two or more jobs have the same index value), select first the job that has the lowest release date among the tie jobs. Also, if there is a further tie in the release date, arrange the jobs in the order of their increasing processing time. If there are still ties, break ties arbitrarily.
STEP 4: Select the job in the first position from Job_Set_D, add this job to Job_Set_B and
remove same job from Job_Set_C.

STEP 5: Compute $\Delta W_j = (p_i+r_i) - R_j$ for all
the remaining jobs in Job_Set_D. Where R_j is the release date of each of the remaining jobs in Job_Set_D, j=2, 3, …, n; i = 1,2,…, n-1.

STEP 6: If ΔW_j is negative or zero, set
$\Delta W_j = \Delta W_i$

STEP 7: Re -arrange the remaining jobs in
Job_Set_D in the order of increasing ΔW_j. If there is a tie in ΔW_j, arrange the tie jobs in the increasing order of their release dates. Also, if there is a further tie in the release date; arrange the tied jobs in the increasing order of their processing times. If there are still ties, break the ties arbitrarily.
STEP 8: Append Job_Set_D to Job_Set_B.
STEP 9: Stop

3.2 Data Analysis

A total of 20 problem sizes ranging from 3 to 100 jobs and 50 problem instances under each problem size were randomly generated (1000 problems in all). The processing times (p_i) of jobs were randomly generated (using random number generator in Microsoft Visual Basic 6.0) with values ranging between 1 and 100 inclusive. Also, the ready times (r_i) of jobs were randomly generated with values ranging between 0 and 49 inclusive.

Coding was carried out in Microsoft Visual Basic 6.0. The program computes the value of total flowtime (F_{tot}) obtained by each solution method and each problem instance. Also computed was the execution time taken by each solution. The Statistical Analysis System (SAS version 9.2) was used to carry out detailed statistical analysis. The hardware used for the experiment had a 1.87 GHz P6000 Intel CPU with 4 GB of main memory.

The mean value of total flowtime obtained by the various solution methods over the 50 problem instances solved under each problem size was computed using the General Linear Models (GLM) procedure in SAS. The GLM procedure was also used to carry out the test of means (t-tests). The t-test was carried out in order to determine whether or not the differences observed in the mean value of total flowtime obtained by the solution methods are statistically significant. In order to measure the tendency of the solution methods to obtain best and worse results, the percentage of time each solution method obtains best and worse results were computed using the Summary procedure in SAS. Also, the approximation ratio of each solution methods was computed. The results obtained are presented and discussed in section 4.

4. RESULTS AND DISCUSSIONS

In order to measure the effectiveness of the solution methods, the value of the total flowtime was computed for each solution method, each problem size and problem instance. The mean value of the total flowtime obtained by the various solution methods over the 50 problem instances solved under the 20 different problem sizes ranging from 3 to 100 jobs is shown in Table 1. For a minimization problem, the smaller the value of the total flowtime, the better the solution method. Based on the minimum mean value of the total flowtime, a ranking order of BB, MPSW, KSA2 and KSA1 was obtained for all the problem sizes considered ($3 \leq n \leq 100$) (Table 1).

In order to determine whether or not the differences observed in the mean value of the total flowtime obtained by the solution methods are statistically significant, the test of means (t-test) was carried out and the results obtained are

summarized in Table 2. The differences observed in the mean values of the total flowtime obtained by the four solution methods (BB, MPSW, KSA1 and KSA2) are not significant at 5% level for all the problem sizes considered (Table 2).

In order to compare the performances of MPSW, KSA1 and KSA2 algorithms with that of the BB method, approximation ratios (MPSW/BB, KSA1/BB and KSA2/BB) were computed and plotted (Figure 1). The MPSW algorithm performed closest to the BB, followed by KSA2 while the KSA1 lag behind (Figure 1). The overall approximation ratio is shown in Table 3. The MPSW, KSA2 and KSA1 algorithms when compared with the BB method have approximation ratios 1.0099 (indicating that with respect to effectiveness the MPSW algorithm is about 0.0099% worse than BB method), 1.0191 (indicating that with respect to effectiveness the KSA2 algorithm is about 0.0191% worse than BB method) and 1.0201 respectively (indicating that with respect to effectiveness the KSA1 algorithm is about 0.0201% worse than BB method) (see Table 3).

To further examine the internal performances of the MPSW, KSA1 and KSA2 solution methods, the percentage of time each solution method obtained the best and worse results was computed (see Tables 4 and 5). The MPSW gave the best results for all the considered problem sizes, indeed yielding 100% in instances involving 7, 80 and 100 jobs (Table 4). Thus, excluding the BB (which is optimal), the MPSW has the tendency to produce best results compared to KSA1 and KSA2. Generally, the KSA1 algorithm has the tendency to produce worse results compared to KSA2 and MPSW (Table 5).

In order to measure the efficiency of the solution methods, the execution time (seconds) taken by each method to obtain a solution to an instance of a problem was computed, plotted and are shown in Figures 2 and 3. As expected, the BB being an implicit enumeration method was the slowest followed by the MPSW (Figure 2). The KSA1 and KSA2 algorithms are very fast. It is observed that the BB and MPSW begin to exhibit exponential time complexity function when the number of jobs exceeds 40 and 70 respectively (Figures 2 and 3). To determine whether or not the differences observed in the mean execution time taken by the solution methods are statistically significant, the test of means (t-test) was carried out and the results obtained summarized in Tables 6 and 7.

The mean time taken by the KSA1 is not significantly different at 5% level from that of the KSA2 algorithm for all the problem sizes considered ($3 \leq n \leq 100$) (Tables 6 and 7). Also, the mean time taken by the MPSW is not significantly different at 5% level from that of the BB for $3 \leq n \leq 25$ problems (Tables 6). However, the mean time taken by the KSA1 and KSA2 algorithms are significantly different at 5% level from that of the MPSW and BB methods (indicating that both KSA1 and KSA2 algorithms are faster than both MPSW and BB) for all the problem sizes considered ($3 \leq n \leq 100$) (Tables 6 and 7). The mean time taken by the MPSW is significantly different (faster) at 5% level from that of BB for $30 \leq n \leq 100$ (Table 7).

Table 1 Mean value of total flow time by problem sizes and solution methods

Problem	Mean of total flow time			
Size	BB	KSA1	KSA2	MPSW
3x1	253.74	256.26	256.26	254.82
4x1	427.28	442.56	441.62	432.64
5x1	554.80	577.54	577.30	565.62
6x1	797.26	820.16	818.70	808.78
7x1	1111.14	1141.80	1141.80	1127.82
8x1	1346.04	1384.12	1382.54	1362.36
9x1	1680.66	1739.62	1738.72	1719.68
10x1	1896.40	1950.98	1950.44	1923.84
11x1	2386.36	2465.50	2465.06	2438.84
15x1	4410.72	4509.24	4499.04	4469.64
20x1	7131.98	7271.92	7257.78	7211.66
25x1	11187.72	11369.34	11356.02	11287.74
30x1	16368.00	16575.20	16558.06	16443.70
40x1	28197.16	28567.22	28478.46	28351.68
50x1	43266.72	43655.86	43582.84	43416.34
65x1	72809.42	73495.42	73454.82	73034.20
70x1	84441.62	85177.96	85101.02	84700.80
80x1	111198.14	112156.78	112015.98	111490.72
90x1	141296.58	142302.42	142289.34	141617.98
100x1	172975.04	174152.22	174176.76	173297.92

Sample size=50

Table 2 Test of means (probability values) of total flow time for $3 \leq n \leq 100$ problems
Solution Methods

Solution Methods	BB	KSA1	KSA2	MPSW
BB	-	>0.5x	>0.50x	>0.50x
KSA1	>0.50x	-	>0.50x	>0.50x
KSA2	>0.50x	>0.50x	-	>0.50x
MPSW	>0.50x	>0.50x	>0.50x	-

Note x indicate non significant result at 5% level; Sample size = 50
\- indicate not necessary

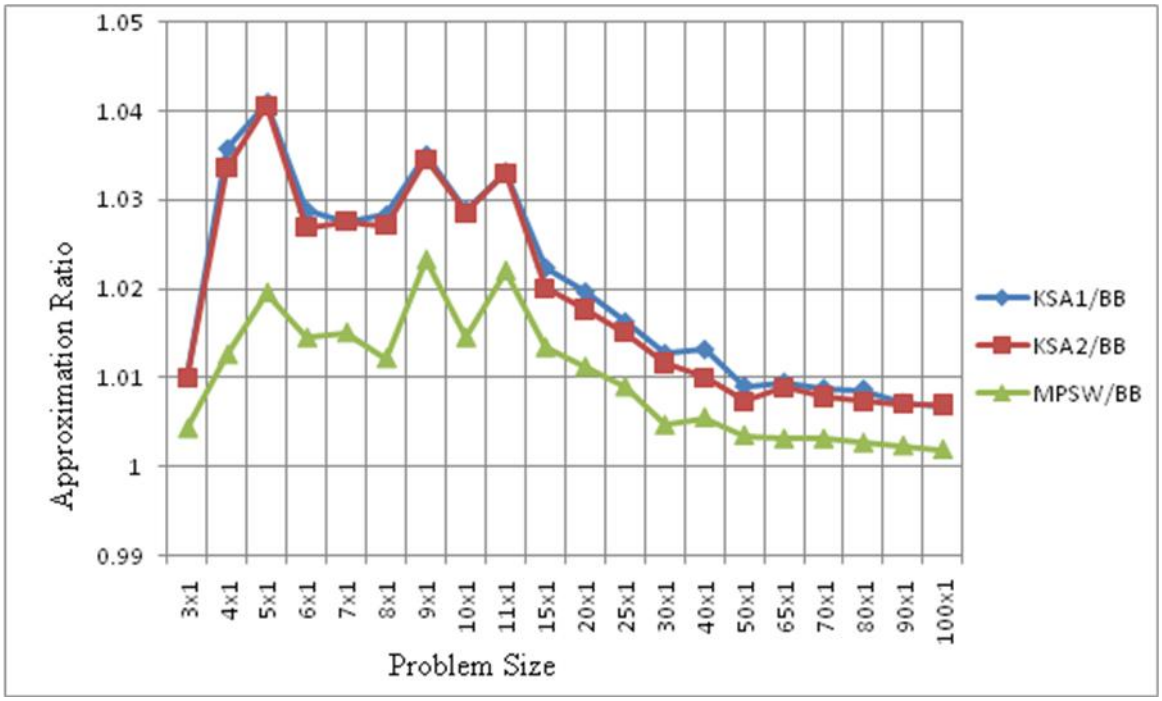

Figure 1 Approximation Ratios (KSA1/BB, KSA2/BB and MPSW/BB) by problem sizes .

Table 3 Overall approximation ratios of total flowtime by solution methods

Solution Methods	Overall approximation ratio
KSA1/BB	1.0201
KSA2/BB	1.0191
MPSW/BB	1.0099

Sample size = 1000

Table 4 Percentage of time solution methods obtain best results by problem sizes

Problem Size	Percentage (%) of time best result was obtained MPSW	KSA1	KSA2
3x1	12	0	0
4x1	56	0	0
5x1	56	0	0
6x1	74	0	0
7x1	100	0	0
8x1	94	0	0
9x1	88	2	0
10x1	90	2	2
11x1	98	0	0
15x1	92	4	0
20x1	80	4	6
25x1	86	0	4
30x1	98	2	0
40x1	96	2	2
50x1	88	4	2
65x1	98	0	2
70x1	98	0	0
80x1	100	0	0
90x1	92	2	2
100x1	100	0	0

Sample size=50

Table 5 Percentage of time solution methods obtain worse results by problem sizes

Problem Size	Percentage (%) of time worse result was obtained MPSW	KSA1	KSA2
3x1	0	0	0
4x1	6	2	0
5x1	4	2	0
6x1	4	4	4
7x1	4	0	0
8x1	4	6	2
9x1	8	6	4
10x1	4	10	8
11x1	2	8	6
15x1	4	14	8
20x1	8	18	12
25x1	6	14	14
30x1	0	22	20
40x1	0	50	14
50x1	8	44	16
65x1	0	40	30
70x1	2	58	20
80x1	0	66	24
90x1	4	38	40
100x1	0	46	46

Sample size=50

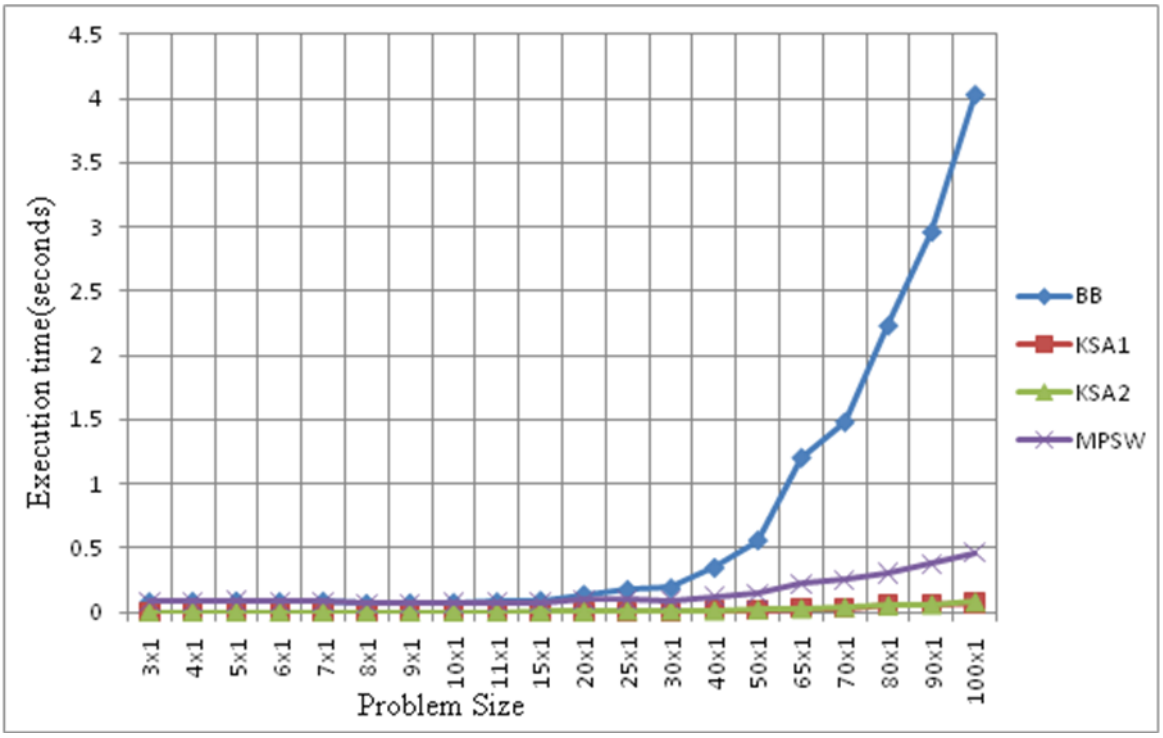

Figure 2 Comparison of the execution time (seconds) taken by four solution methods and problem sizes

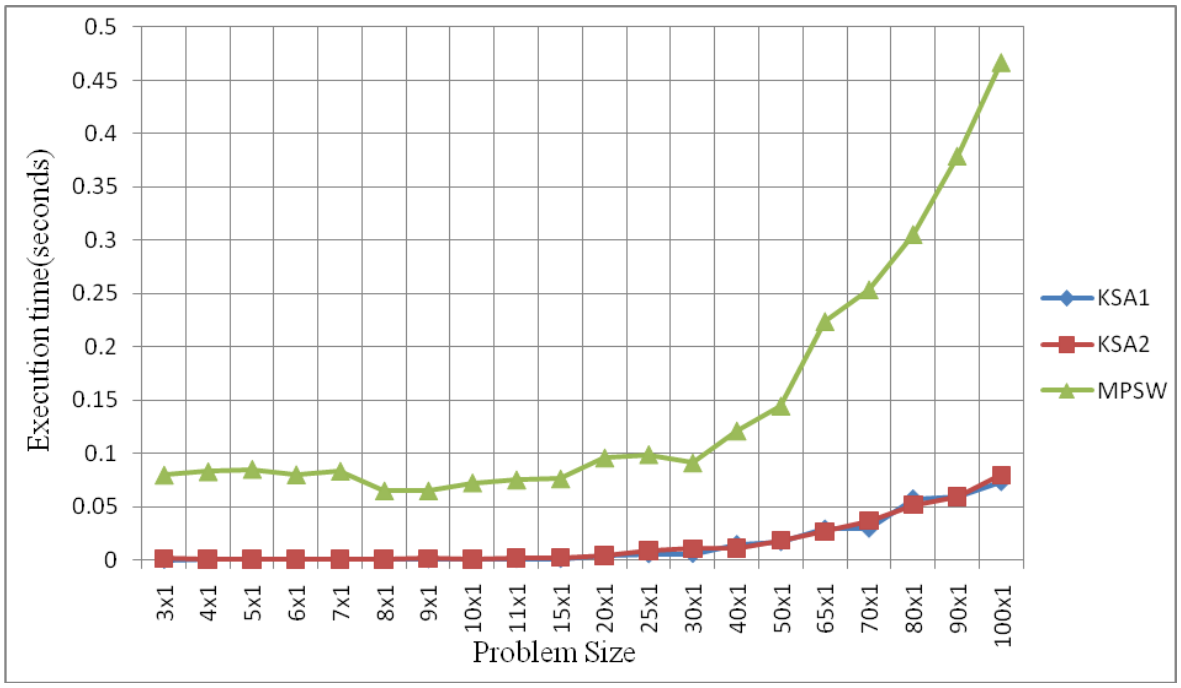

Figure 3 Comparison of the execution time (seconds) taken by three solution methods and problem sizes

Table 6 Test of means (probability values) of execution time for $3 \leq n \leq 25$ problems

	Solution Methods			
Solution Methods	**BB**	**KSA1**	**KSA2**	**MPSW**
BB	-	<0.050*	<0.050*	>0.50x
KSA1	<0.050*	-	>0.50x	>0.50x
KSA2	<0.050*	>0.50x	-	>0.50x
MPSW	>0.50x	<0.050*	<0.050*	-

Note * indicates significant result at 5% level; Sample size = 50
x indicates non significant result at 5% level
- indicates not necessary

Table 7 Test of means (probability values) of execution time for $30 \leq n \leq 100$ problems

	Solution Methods			
Solution Methods	**BB**	**KSA1**	**KSA2**	**MPSW**
BB	-	<0.005*	<0.005*	<0.005*
KSA1	<0.005*	-	>0.50x	>0.50x
KSA2	<0.005*	>0.50x	-	>0.50x
MPSW	<0.005*	<0.005*	<0.005*	-

Note * indicates significant result at 5% level; Sample size = 50
x indicates non significant result at 5% level
- indicates not necessary

5. CONCLUSIONS

We have explored the scheduling problem of minimizing the sum of flowtime (total flowtime) on a single machine with release dates. Two approximation algorithms (KSA1 and KSA2) were proposed for solving the problem. The two algorithms were compared with the MPSW algorithm selected from the literature. In order to further measure the performance of the three solution methods (KSA1, KSA2 and MPSW), they were compared with the Branch and Bound (BB) method.

Performance evaluations were based on both effectiveness (closeness of the value of the composite objective function to the optimal) and efficiency (how fast solution can be obtained i.e. a measure of execution speed). Experimental results, based on effectiveness, show that the MPSW algorithm outperformed both KSA1 and KSA2 algorithms as well as performed closest to the BB for all the problem sizes considered. Based on effectiveness, the KSA2 algorithm outperformed the KSA1 algorithm in many of the problem sizes while it performed as good as KSA1 in some of the problem sizes considered (3 and 7 jobs). However, with respect to efficiency, the KSA2 and KSA1 algorithms outperformed the MPSW algorithm for all the problem sizes considered. Also, the KSA2 was found to be as efficient as the KSA1 (i.e. none was consistently faster than the other).

Therefore, based on effectiveness, the MPSW algorithm is recommended for the scheduling problem of minimizing the total flowtime on a single machine with release dates while, based on efficiency, the KSA2 algorithm is recommended for the scheduling problem of minimizing the total flowtime on a single machine with release dates.

6. REFERENCES

[1] Ahmadi, R. H. & Bagchi, U. 1990. Lower bounds for single-machine scheduling problems. *Naval Research Logistics*, 37, pp. 967–979.

[2] Baker, K. R. 1974. Introduction to sequencing and scheduling. New York: Wiley.

[3] Chandra, R. 1979. On n|1| $\overline{F}$ dynamic deterministic problems. *Naval Research Logistics*, 26, pp. 537–544.

[4] Chu, C. 1992. A branch-and-bound algorithm to minimize total flow time with unequal release dates. *Naval Research Logistics*, 39, pp. 859–875.

[5] Deogun, J. S. 1983. On scheduling with ready times to minimize mean flow time. *Computer Journal*, 26, pp. 320–328.

[6] Dessouky, M. I., & Deogun, J. S. 1981. Sequencing jobs with unequal ready times to minimize mean flow time. *SIAM Journal on Computing*, 19, pp. 192–202.

[7] Graham, R.L., Lawler, E.L., Lenstra, J.K. & Rinnooy Kan, A.H.G. 1979. Optimization and approximation in deterministic sequencing and scheduling: A survey, *Annals of Discrete Mathematics*, 5, pp. 287-326.

[8] Guo, Y., Lim, A., Rodrigues, B. & Yu, S. 2004. Minimizing total flow time in single machine environment with release time: an experimental analysis, *Computers & Industrial Engineering*, 47, pp. 123–140.

[9] Karger, D. Stein, C. & Wein, J. 1997. Scheduling Algorithms, A chapter written for the CRC Handbook on Algorithms (Boca Raton 1997).

[10] Lenstra, J. K., Rinnooy Kan, A. H. G., & Brucker, P. (1977). Complexity of machine scheduling problems. Annual Discrete Mathematics, 1, pp. 343–362.

[11] Leonardi, S., & Raz, D. 1997. Approximating total flow time on parallel machines, ACM Symposium on Theory of Computing.

[12] Oyetunji, E. O. 2009. Some common performance measures in scheduling problems, *Research Journal of Applied Sciences, Engineering and Technology*, 1(2), pp. 6-9.

[13] Oyetunji, E.O. & Oluleye, A.E. 2007. Heuristics for minimizing total completion time on single machine with release time, *Advanced Materials Research*, Trans Tech Publications Ltd., Switzerland, 18-19, pp. 347-352.

[14] Phillips, C., Stein, C.,& Wein, J. 1998. Minimizing average completion time in the presence of release dates. *Mathematical Programming B*, 82, pp. 199–224.

[15] Smith, W. E. 1956. Various optimizers for single state production. *Naval Research Logistics Quarterly*, 3, pp.56–66.