academicJournals

Vol. 5(7), pp. 207-216, September 2013
10.5897/ JETR-2013-0295
ISSN 2006-9790 © 2013 Academic Journals
http://www.academicjournals.org/JETR

Journal of Engineering and Technology Research

Full Length Research Paper

A comparative study of meta-heuristics for identical parallel machines

M.O. Adamu[1]* and A. Adewunmi[2]

[1]Department of Mathematics, University of Lagos, Lagos, Nigeria.
[2]School of Computer Science, University of Kwazulu-Natal, Durban, South Africa.

Accepted 15 February, 2013

This paper considers the scheduling problem of minimizing the weighted number of early and tardy jobs on identical parallel machines, $Pm||\sum w_j(U_j + V_j)$. This problem is known to be NP complete and finding an optimal solution is unlikely. Six meta-heuristics including hybrids are proposed for solving the problem. The meta-heuristics considered are genetic algorithm, particle swarm optimization and simulated annealing with their hybrids. A comparative study that involves computational experiments and statistical analysis are presented evaluating these algorithms. The results of the research are very promising.

Key words: Parallel machine, heuristics, just-in-time, meta-heuristics, NP-complete.

INTRODUCTION

Scheduling Just-In-Time (JIT) jobs is of great importance in both manufacturing and service industries. Production wastages are reduced and profitability is improved when JIT is applied. Its application cuts across medical, machine environment, distribution network and other environments. In this paper, we consider the comparative study of various heuristics for scheduling weighted jobs on identical parallel machines. The objective is to minimize the weighted number of early and tardy jobs on identical parallel machines.

During the past few decades, a considerable amount of work has been done on scheduling on multiple machines to minimize the number of tardy jobs (Adamu and Adewunmi, 2012b) and on single machine (Adamu and Adewunmi, 2012c). Garey and Johnson (1979) have shown our problem to be NP-complete and finding an optimal solution appears unlikely. Using the three-field notation of Graham et al. (1979), the problem is represented as $Pm||\sum w_j(U_j + V_j)$. Scheduling to minimize the (weighted) number of tardy jobs have been considered by Ho and Chang (1995), Süer et al. (1993), Süer (1997), Süer et al. (1997), Van der Akker (1999), Chen and Powell (1999), Liu and Wu (2003), and M'Hallah and Bulfin (2005). Sevaux and Thomin (2001) addressed the NP-hard problem to minimize the weighted number of late jobs with release time ($P|r_j|\sum w_jU_j$). They presented several approaches for the problem including two MILP formulations for exact resolution and various heuristics and meta-heuristics to solve large size instances. They compared their results to that of Baptiste et al. (2000) which performed averagely better. Baptiste et al. (2000) used a constraint based method to explore the solution space and give good results on small problems ($n < 50$). Dauzère-Pérès and Sevaux (1999) determined conditions that must be satisfied by at least one optimal sequence for the problem of minimizing the weighted number of late jobs on a single machine. Sevaux and Sörensen (2005) proposed a variable neighbourhood search (VNS) algorithm in which a *tabu* search algorithm is embedded as a local search operator. The approach was compared to an exact method by Baptiste et al. (2000). Li (1995) addressed the $P|$agreeable due dates $|\sum U_j$ problem. Where the due dates and release times are assumed to be agreeable. A

*Corresponding author. E-mail: madamu@unilag.edu.ng.

heuristic algorithm is presented and a dynamic programming lower bounding procedure developed. Hiraishi et al. (2003) addressed the non preemptive scheduling of *n* jobs that are completed exactly at their due dates. They showed this problem is polynomially solvable even if positive set-up is allowed. Sung and Vlach (2001) showed that when the number of machines is fixed, the weighted problem considered by Hirashi et al. (2003) is solvable in polynomial time (exponential in the number of machines) no matter whether the parallel machines are identical, uniform or unrelated. However, when the number of machines is part of the input, the unrelated parallel machine case of the problem becomes strongly NP-hard. Lann and Mosheiov (2003) provided a simple greedy O(n log n) algorithm to solve the problem of Hiraishi et al. (2003) greatly improving in the time complexity. Čepek and Sung (2005) considered the same problem of Hiraishi et al. (2003) where they corrected the greedy algorithm of Lann and Mosheiov (2003) that was wrong and presented a new quadratic time algorithm which solved the problem. Adamu and Abass (2010) proposed four greedy heuristics for the $Pm||\sum w_j (U_j + V_j)$ problem and extensive computational experiments performed. Janiak et al. (2009) studied the problem of scheduling n jobs on m identical parallel machines, in which for each job a distinct due window is given and the processing time is unit time to minimize the weighted number of early and tardy jobs. They gave an $O(n^5)$ complexity for solving the problem $(Pm|p_j = 1 |\sum w_j(U_j + V_j)$. They also consider a special case with agreeable earliness and tardiness weights where they gave on $O(n^3)$ complexity $(Pm|p_j = 1, r_j,$ agreeable ET weights$|\sum w_j(U_j + V_j))$. Adamu and Adewunmi (2012a) compared the heuristics of Adamu and Abass (2010) with some metaheuristics.

PROBLEM FORMULATION

A set of independent jobs $N = \{1,2, . . . , n\}$ has to be processed on *m* parallel identical machines, which are simultaneously available from time zero, each having an interval rather than a point in time, called due window of the job. The left end and the right end of the window are respectively called the earliest due date (that is, the instant at which a job becomes available for delivery), and the latest due date (that is, the instant by which processing or delivery of a job must be completed). There is no penalty when a job is completed within the due window, but for earliness or tardiness, penalty is incurred when a job is completed before the earliest due date or after the latest due date. Each job $j \in N$ has a processing time p_j, earliest due date a_j, latest due date d_j and a weight w_j. it is assumed that there is no preemptions and only one job is allowed to be processed on a given machine at any given time. For any schedule S, let t_{ij} and $C_{ij}(S) = t_{ij} + p_j$ represent the actual start time on a given machine and completion time of job j on machine i, respectively. Job j is said to be early if $C_{ij}(S) < a_j$, tardy if $C_{ij}(S) > d_j$ and on-time if $a_j \le C_{ij}(S) \le d_j$. For any job j, the weighted number of early and tardy jobs (Liu and Wu, 2003)

$$w_j U_j = w_j \operatorname{int}\left\{\frac{1}{2} sign[C_{ij}(S) - p_j] + \frac{1}{2}\right\}$$

Where we define that

$$sign[C_{ij}(S) - p_j] = \begin{cases} 1, & if\ a_j > C_{ij}(S) \quad OR \quad C_{ij}(S) > d_j \\ -1, & a_j > C_{ij}(S) \quad OR \quad C_{ij}(S) > d_j \end{cases}$$

and that int is the operation of making an integer. Obviously,

$$U_j = \begin{cases} 1, & if\ a_j > C_{ij}(S) \quad OR \quad C_{ij}(S) > d_j \\ 0, & a_j > C_{ij}(S) \quad OR \quad C_{ij}(S) > d_j \end{cases}$$

Therefore, the scheduling problem of minimizing the weighted number of tardy jobs on identical parallel machines can be formulated as G.

$$G = \sum_{i=1}^{m} \sum_{j=1}^{n} w_j U_j = \sum_{i=1}^{m} \sum_{j=1}^{n} w_j \operatorname{int}\left\{\frac{1}{2} sign[C_{ij}(S) - p_j] + \frac{1}{2}\right\} \quad (1)$$

$$\text{Min } G = \sum_{i=1}^{m} \sum_{j=1}^{n} w_j U_j = \min \sum_{i=1}^{m} \sum_{j=1}^{n} w_j \operatorname{int}\left\{\frac{1}{2} sign[C_{ij}(S) - p_j] + \frac{1}{2}\right\} \quad (2)$$

HEURISTIC AND META-HEURISTICS

Greedy heuristic

Adamu and Abass (2010) have proposed four greedy heuristics which attempt to provide near optimal solutions to the parallel machine scheduling problem. In this paper the fourth heuristic (DO2) would be use. It entails sorting the jobs according to their latest due date (that is, latest due time - processing time) and ties broken by the highest weighted processing time is used (that is, weight / processing time).

Results of these greedy heuristics are encouraging; however it will be further investigated whether using meta-heuristics and their hybrids can achieve better results.

Genetic algorithm

Genetic algorithms (GAs) are one of the best known meta-heuristics for solving optimization problems. GAs are loosely based on evolution in nature and use strategies such as survival of the fittest, genetic crossover and mutation. Since GAs usually have a high performance and also use a population based technique, it was decided to investigate their comparative performance with the greedy heuristics.

Problem representation

Deciding on a suitable representation is one of the most important aspects of a GA. It was decided that each job would be fixed to a gene in the chromosome – implying that the chromosome has length n (where n is the number of jobs). Each gene would also have a machine number (the number of the machine to which the job will be assigned) and an order (a value between 1 and n representing the order in which jobs assigned to the same machine will be executed). Genetic operators would then need to be applied to both the machine number and the order.

Algorithm

A basic pseudo code of the genetic algorithm found in Adamu and Adewunmi (2012a) was used.

Fitness function

The fitness function calculates the sum of the weights of jobs which could not be assigned onto any of the machines so that they would finish within the earliest due and latest due dates. For each machine, jobs which are assigned to it are placed in a priority queue (which bases priority on their respective order). Each job is then removed from the queue and placed on the machine. If the job was to finish early, then it would be scheduled to begin later (at earliest due date -processing time) in order to avoid the earliness penalty. However, if the job was to finish past the end time, then it would not be scheduled at all and instead would have its weight added to the total penalty (fitness). One final, important aspect to note is that a lower fitness function implies a better performance.

Genetic operators

Genetic algorithms have a large number of operators available to them as well as different implementations of the operators which may be useful in different situations. In the initial version of the GA, the following operators were used: 1-point crossover for machines, conventional mutation for machines (that is, choose a random machine between 0 and m-1 inclusive), swap mutation for the execution order (since naturally this is permutation based) and tournament selection. However, since there are no guarantees that these operators allowed for the best performance, further experimentation with variations of these operators was performed. More details will be given subsequently.

Particle swarm optimization (PSO)

Particle swarm optimization was chosen to attempt to solve the parallel machine scheduling problem. It is a population based technique derived from the flocking behaviour of birds which relies on both the particle's best position found so far as well as the entire population's best position to get out of local optimums and to find the global optimum. PSO is appropriate to use for parallel machine scheduling because not much is known about the solution landscape and so PSO may be useful to get out local optimums to find the global optimum.

Problem representation

The PSO algorithm requires that a representation of the solution (or encoding of the solution) is chosen. Each particle will be instances of the chosen representation. A complication is that PSO works in the continuous space whereas the scheduling problem is a discrete problem. Thus, a method is needed to convert from the continuous space to the discrete space. The representation is as follows:

(i) Each particle contains a number between 0 (inclusive) and the number of machines (exclusive). This number represents the machine on which the particle is scheduled and is simply truncated to convert to the discrete space.
(ii) Each particle contains a number between 0 (inclusive) and 1 (exclusive). This number represents the order of scheduling relative to the other particles on the same machine where a lower number indicates that that job will be scheduled before the jobs with higher numbers.

Algorithm

A basic pseudo code of the PSO found in Adamu and Adewunmi (2012a) was used.

Fitness function

Finally, a method is needed to convert the encoding into a valid schedule (this is performed when calculating the fitness).

This is performed by separating the jobs into groups based on the machine to which they are assigned. Within a group, the jobs are sorted by their order parameter and organized into a queue. The schedule for a particular machine is then formed by removing jobs from the queue and scheduling them as early as possible without breaking the earliness constraint. The weights of jobs that cannot be scheduled are totaled as the fitness of the solution (which would ideally be as small as possible).

Simulated annealing

Simulated annealing (SA) was chosen as a meta-heuristic which could solve the parallel machine scheduling problem. Simulated annealing is based on real-life annealing, where the heating of metals allows for atoms to move from their initial position and the cooling allows for the atoms to settle in new optimal positions. SA is not a population based heuristic – thus only one solution is kept at any one stage. Since SA should result in less operations being performed with respect to a population based technique, execution times may be quicker. It is this reason why SA was chosen for investigation.

It should also be noted that simulated annealing will in all likelihood achieve better results than a simple hill-climbing technique. This is because SA can take downward steps (that is, accept worse solutions) in order to obtain greater exploration. Thus, it is less likely to become stuck in a local minimum (a very real problem given the complex solution space).

Problem representation

The representation is remarkably similar to that used in the GA. A solution consists of n elements (where n is the number of jobs). Each element has a specific job as well as the machine onto which it will be assigned and the order of assignment. Perhaps the major difference between them is that the GA has a population of solutions (chromosomes) whereas SA focuses on a single solution.

Algorithm

A basic algorithm used in the SA [found in Adamu and Adewunmi (2012a)] technique:

Fitness function

Since, the solution is represented in virtually the exact same manner as a chromosome in the GA and a particle in PSO, the fitness function is calculated in the same manner. That is, jobs pertaining to a particular machine are placed in a priority queue before being assigned onto the machine. Those which cannot be assigned contribute towards the penalty.

Operators

Although, simulated annealing does not really have operators (in the sense of a GA having genetic operators), the SA algorithm does has to select a neighbor. The particular neighbor selection strategy that is used updates only a single element of the solution. The element is given a new randomly chosen machine and a new order (done by swapping with the order of another randomly chosen element). By allowing for a high level of randomness when selecting the neighbor, it will be ensured that good exploration will be achieved and that a local best is not found too early.

COMPUTATIONAL ANALYSIS AND RESULTS

Date generation

The program was written in Java using Eclipse. It actually consists of a number of programs, each one implementing a different type of solution. The output of each of these programs gives the final fitness after the algorithm has been performed and the time in milliseconds that the algorithm took to run.

The heuristics were tested on problems generated with 100, 200, 300 and 400 jobs similar to Adamu and Abass (2010), Ho and Chang (1995), Baptiste et al. (2000), and M'Hallah and Bulfin (2005). The number of machines was set at levels of 2, 5, 10, 15 and 20. For each job j, an integer processing time p_j was randomly generated in the interval (1, 99). Two parameters, k1 and k2 (levels of Traffic Congestion Ratio) were taken from the set {1, 5, 10, 20}. For the data to depend on the number of jobs *n*, the integer earliest due date (a_j) was randomly generated in the interval (0, n / (m * k1)), and the integer latest due date (d_j) was randomly generated in the interval ($a_j + p_j$, $a_j + p_j$ + (2 * n * p) / (m * k2)).

For each combination of n, k1 and k2, 10 instances were generated, that is, for each value of n, 160 instances were generated with a weight randomly chosen in interval (1, 10) for 8000 problems of 50 replications. The meta-heuristics were implemented on a Pentium Dual 1.86 GHz, 782 MHz, and 1.99 GB of Ram. The following meta-heuristics were analyzed GA, PSO, SA, GA Hybrid, PSO Hybrid, PSOGA Hybrid and SA Hybrid.

Improvements

Genetic algorithms are different from many other meta-heuristics in that they have different genetic operators which can be tried and tested – rather than simply changing parameters. The original GA which was tested used 1-point crossover, random mutation for machines, swap mutation for order and tournament selection. It was decided to try other combinations of operators in order to see if performance could be increased. For this reason, roulette-wheel selection, uniform crossover and insert mutation (for order) were all programmed. A user would then be able to choose any combination of operators to use for their own GA. More information on the optimal combination of genetic operators will be mentioned subsequently in the parameters.

Greedy hybrids

Once the meta-heuristics (GA, PSO and SA) had been programmed, it was thought that improvements on them could potentially be made if they somehow included aspects or features from the greedy heuristic used by Adamu and Abass (2010). It was clear from the works of Adamu and Abass (2010) that the key to the greedy heuristics was in the order in which jobs were assigned to machines. So the mechanisms of ordering in DO2 needed to be incorporated in the meta-heuristics (GA, PSO, SA).

To implement the hybridization in the 3 meta-heuristics, the order field was removed from Gene, Dimension and Element respectively. Also, any code in Chromosome, Particle and Solution which dealt with the order (for example, swap mutation in Chromosome) was removed.

Parameters

For each solutions strategy, there are a number of different parameters that affect the performance of the algorithm such as population size, mutation rate, initial temperature, etc. These parameters needed to be experimentally determined and so the algorithms were run manually on a subset of all the testing data in order to determine the optimal parameters. This involved experimenting with the full range of each parameter and recording and tabulating the results achieved. The combination of parameters that gave the best performance was selected as the optimal parameters.

The optimal parameters for the genetic algorithm are:

(i) A population size of 10.
(ii) Random mutation (for machines) used at a rate of 0.01.
(iii) Swap mutation (for order) used at a rate of 0.01.
(iv) Uniform crossover at a rate of 0.5.
(v) Tournament selection with a k set at 40% of the population size.
(vi) The number of iterations of the algorithm was set at 2000.

Further to the above parameters, the genetic algorithm hybrid achieved best results when hybridized with the DO2 greedy heuristic.

The optimal parameters for particle swarm optimization are:

(i) A population size of 50.
(ii) A w (momentum value) of 0.3.
(iii) A c1 of 2.
(iv) A c2 of 2.
(v) The number of iterations of the algorithm was set at 2000.

Further to the above parameters, the particle swarm optimization hybrid achieved best results when hybridized with the DO2 greedy heuristic.

The optimal parameters for simulated annealing are:

(I) An initial temperature of 25.
(ii) A final temperature of 0.01.
(iii) A geometrical decreasing factor (beta) of 0.999.

Further to the above parameters, the simulated annealing hybrid achieved best results when hybridized with the DO2 greedy heuristic.

DISCUSSION

In this part of the work, the results of the algorithms are shown, including the hybridizations. In the four columns shown in Table 1, each cell consists of two numbers. The top number is the weight of the schedule that is produced, averaged over 50 runs. The bottom number is the average time in milliseconds that the algorithm takes to complete.

Also included are four charts each for the performance of the meta-heuristics in relation to the penalty (Figure 1) and time (Figure 2) for N= 100, 200, 300 and 400. Figure 1 compares the relative performance (penalty) of each of the 6 algorithms compared to the number of machines used. Again, four charts are given to show the computational times of the meta-heuristics for various values of N. It should be clear from both the Table 1 and the charts that the Simulated Annealing Hybrid (SAH) out performed the other meta-heuristics in almost all points and the over all lowest time averagely less than a second. It was observed the various hybrids performed better than their meta-heuristic without it. It further proves the effectiveness of hybridization on the meta-heuristics.

The Genetic algorithm (GA) performed worst compared to other meta-heuristics in all of the categories considered for all N jobs and M machines. The GA time is averagely 2.8 s, far slower than the SAH – notably because it keeps track of a population of individual solutions. Results show it to be in the region of 2.8 times slower compared to SAH.

The genetic algorithm which is hybridized with DO2 (GAH) achieves better results (Table 1 and Figure 1) compared to the simple genetic algorithm (GA) on all of the test cases. In all cases considered, the GAH outperform the ordinary GA and as the value of N increases the performance rate of GAH over GA widens. For larger values of N the performance of GAH is almost equivalent if not better than SAH. GAH takes on average about 2.77 s. GAH would be ideal for larger values of N where an optimal solution is not readily feasible.

The particle swarm optimization (PSO) and the hybrid PSO (PSOH) produce lower weight compared to the GA. Furthermore, they are far slower than all the meta-heuristics considered (over 14.4 times slower for PSO and 10.5 for PSOH in relation to SAH). This is understandable since PSO is a population-based algorithm so there is a lot of work being done at each step. Hybridizing particle swarm optimization with the DO2 greedy heuristic produces results which are better than PSO for all cases. The PSOH is also about 1.37 times faster than PSO.

The results for simulated annealing (SA) are far better on the average than those GA, PSO and PSOH both in performance of penalty and time (Tables 1 and 2 and Figures 1 and 2). On average, SA takes 1 s to run. However, it is about 2.8, 2.77, 14.4 and 10.5 times quicker than the GA, GAH, PSO and PSOH respectively (Table 5).

Hybridizing simulated annealing with the DO2 greedy heuristic (SAH) produces results that are slightly better than the SA solution for all cases considered. It produces the overall best results among the meta-heuristics in terms of performance in relation to penalty and time. The average timing is a little less than a second.

Further statistical analysis are carried out for both the penalty and timing of the various algorithms. Test of homogeneity of variances, ANOVA test, multiple comparisons test and homogeneous subsets are considered. Tables 2 to 4 are for the penalty performance and time performance. For the penalty performance, it is discovered that the variances of the penalties are not significantly different. Table 2 presents the ANOVA table for penalties. The means of the meta-heuristics are significantly different from one another, that is, they do not have equal means. Due to equality of their variances, subsets of homogeneous groups are displayed in Table 3 using Scheffe's method. Four groups are obtained: group 1—SAH, GAH and SA, group 2—GAH, SA and PSOH, group 3 – PSOH and PSO, and group 4 – GA. These groups are arranged in decreasing order of their effectiveness. The worst among them is the GA. Similarly, for the time performance, Table 4 shows the ANOVA table for the test of equality of the mean time of the meta-heuristics which are also significantly different.

This implies that timings for the various algorithms are not the same. PSO and PSOH have the highest time of 14. 4 and 10.5 s respectively. While the lowest of about 1 s for both SA and SAH.

Conclusion

This paper presents results on scheduling on identical

Table 1. Performance of Meta-heuristics for different N.

	M=2			M=5			M=10			M=15			M=20		
	MIN	**AVE**	**MAX**	**MIN**	**AVE**	**MAX**	**MIN**	**AVE**	**MAX**	**MIN**	**AVE**	**MAX**	**MIN**	**AVE**	**MAX**
								N=100							
GA	593	655.72	730	525	610.68	700	510	558.06	628	458	522.42	557	444	519.22	603
	1906	3404.18	5844	4265	4962.46	5921	4218	4897.44	5937	4235	4942.52	5938	4328	5091.82	6218
GAH	313	374.78	444	257	340.22	397	244	304.78	385	197	261.2	323	202	268.76	309
	2750	2947.88	3266	2640	2826.58	3125	2485	2667.54	2953	2484	2674.44	3078	2843	3048.78	3360
PSO	385	459.9	559	339	482.84	583	352	470.16	559	419	455.74	512	425	477.48	549
	13516	14506.84	29969	13188	14182.52	25578	13047	14038.78	27688	13218	14291.92	26968	13516	14554.38	28578
PSOH	309	374.76	474	289	432.4	511	308	450.56	550	304	438.08	484	342	413.22	472
	10125	10785.3	11531	9750	10357.54	11172	9390	10139.38	13453	9578	10346.24	19688	10735	11605.54	20531
SA	330	378.38	441	294	348.6	417	242	297.02	366	188	247.92	292	216	262.94	317
	421	470.08	547	406	884.64	1375	875	1083.78	1359	907	1092.8	1344	937	1127.8	1531
SAH	342	397.94	473	246	315.14	380	200	258.44	329	167	211.82	274	173	231.52	282
	532	584.36	657	500	529.52	578	453	487.22	563	453	495.32	562	531	574.1	656
								N=200							
GA	535	605.24	717	475	544.34	635	418	482.1	538	387	439.78	535	357	408.18	465
	3297	5051	6188	1843	1982.46	2265	1812	1967.26	2328	1812	1963.82	2156	1859	2011.86	2329
GAH	124	180.22	247	99	163.72	240	92	146.74	220	75	129.24	171	69	107.22	161
	2734	2956.4	3313	2625	2796.28	3078	2453	2650.34	2953	2468	2629.22	2875	2484	2673.74	3015
PSO	285	345.68	410	312	358.06	448	188	361.34	438	225	354.64	420	300	354.52	409
	13406	14314.02	19063	13172	13979.1	15266	13078	13795.56	14829	13172	13949.72	14921	13484	14203.38	15062
PSOH	127	181.36	225	190	257.36	318	184	301.18	344	197	302.32	360	265	303.38	341
	10187	10841.88	16500	9672	10437.5	19250	9422	10144.36	17766	9484	10185.36	18485	9562	10376.2	17328
SA	157	231.2	289	162	206.5	272	93	170.5	230	103	139.48	189	78	108.22	155
	984	1174.36	1438	922	1116.52	1469	875	1074.72	1328	891	1103.5	1390	937	1121.2	1422
SAH	138	190.74	277	93	144.18	210	58	116.5	173	60	91.4	140	35	66.74	115
	547	1232.94	1672	1093	1331.58	1656	453	510.6	1282	431	488.72	578	453	880.02	1375
								N=300							
GA	475	591.46	665	463	520.6	583	386	449.34	540	319	400.3	469	323	371.86	451
	1906	2064.98	2359	1859	1987.52	2438	1813	1961.62	2531	1812	1993.14	2281	1875	2036.58	2922
GAH	33	85.06	146	23	68.24	120	27	61.5	139	25	50.16	95	16	42.72	78
	2750	2964.08	3437	2594	2837.8	4672	2469	2649.78	3735	2453	2657.44	4594	2453	2653.4	3359
PSO	228	304.32	392	210	298.02	383	162	306.24	379	234	306.7	377	165	309.72	384
	13422	14430.36	25984	13234	14209.66	27000	13109	14092.14	28438	13281	14214.64	26313	13453	14439.7	25375

Table 1. Contd.

PSOH	36	84.14	165	100	154.42	211	70	196.2	268	144	209.3	267	167	215.44	264
	10000	10703.48	11547	9656	10202.08	10984	9344	9937.72	10734	9375	9990.48	10766	9516	10167.48	11062
SA	96	164.22	237	92	130.72	195	53	102.34	169	50	79.16	136	33	61.74	87
	984	1185.98	1453	922	1115.34	1375	875	1079.92	1438	906	1079.66	1406	937	1142.32	1407
SAH	23	88.68	171	24	61.22	112	13	43.28	108	12	30.54	65	5	21.24	42
	1219	1445.26	1672	1078	1329.92	1656	1016	1216.54	1547	1000	1200.98	1563	1031	1216.58	1484
								N=400							
GA	483	573.08	668	413	496.28	589	316	424.66	485	308	368.82	483	290	340.28	410
	1906	2056.46	2625	1843	2002.46	3218	1813	1980.1	3125	1828	1994.6	2718	1875	2035.32	2610
GAH	0	28.42	67	0	17.86	45	1	16.04	81	0	9.76	29	0	9	29
	2719	2939.6	3718	2609	2802.28	3453	2422	2609.76	2750	2485	2643.16	4157	2454	2684.06	4500
PSO	204	285.2	362	183	257.7	349	119	262.04	356	187	265.8	330	228	274.32	319
	13406	14165.64	15157	13172	15126.54	27078	13015	14900	36375	13203	15070.32	36969	13468	15490.76	32828
PSOH	2	25.84	56	27	80.52	122	6	116.62	189	32	135.3	195	33	146.64	199
	9937	10654.98	11625	9594	10375.08	18703	9344	11234.38	27438	9266	11292.5	28157	9453	10294.68	18750
SA	82	124.62	184	29	82.3	157	22	58.46	98	12	39.42	64	6	29.5	59
	985	1109	1453	921	1112.82	1547	875	1064.52	1282	907	1106.44	1484	937	1149.38	1500
SAH	1	29.48	84	2	16.44	48	0	11.36	53	0	4.9	19	0	2.62	15
	1219	1459.4	1781	1093	1320.86	1735	1015	1192.86	1453	1016	1188.22	1609	1015	1220.66	1625

Table 2. ANOVA.

Penalty	Sum of squares	df	Mean square	F	Sig.
Between groups	2170218.657	5	434043.731	37.688	0.000
Within groups	1312911.671	114	11516.769		
Total	3483130.328	119			

parallel machines with the objective of minimizing the weighted number of early and tardy jobs. Six meta-heuristics including hybridization are proposed for solving the problem. Extensive computational experiments are performed to analyze these meta-heuristics. It was observed that the simulated annealing hybrid gives the best result both in performance and timing while the genetic algorithm was the worst among them in performance. Further research will focus on comparing these results with optimal solutions and considering other machine environment like

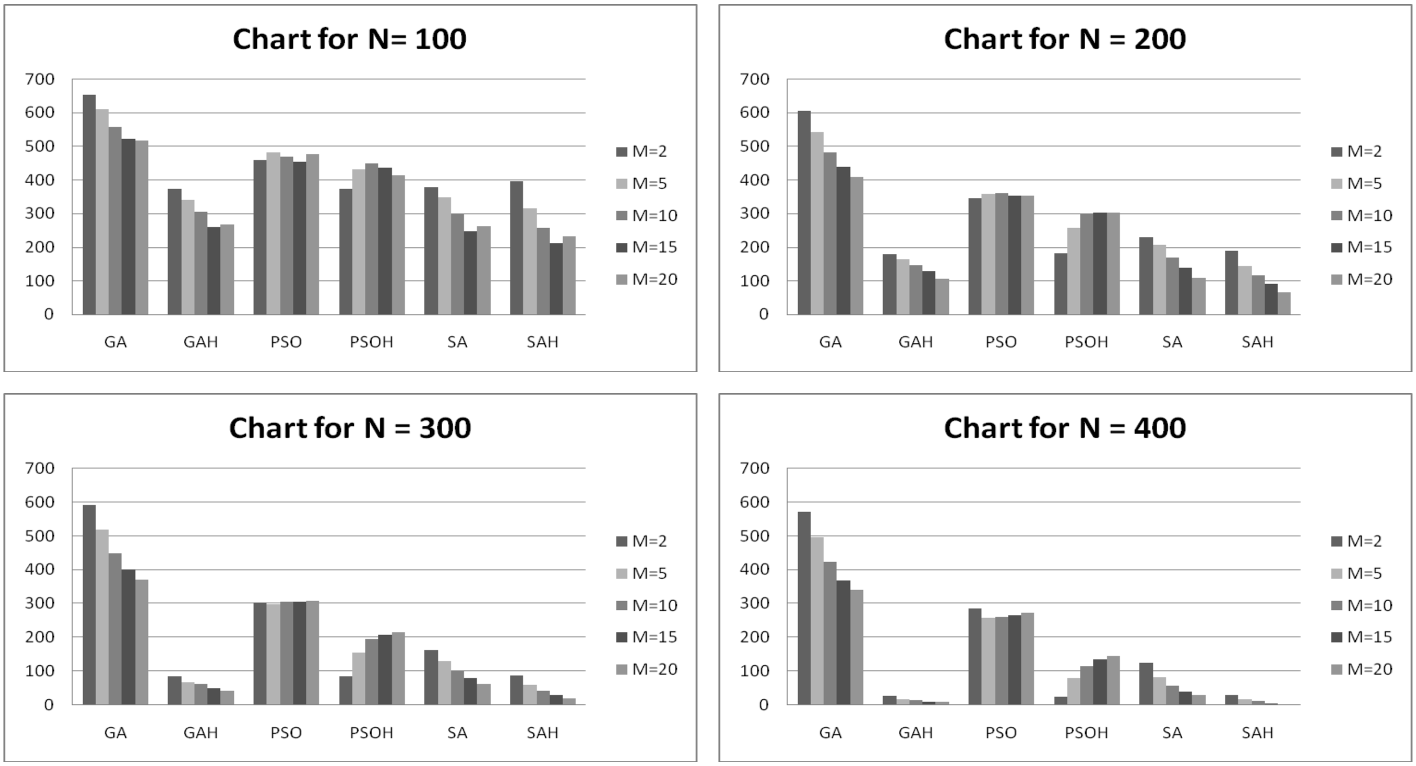

Figure 1. Meta-heuristics performance in relation to penalty.

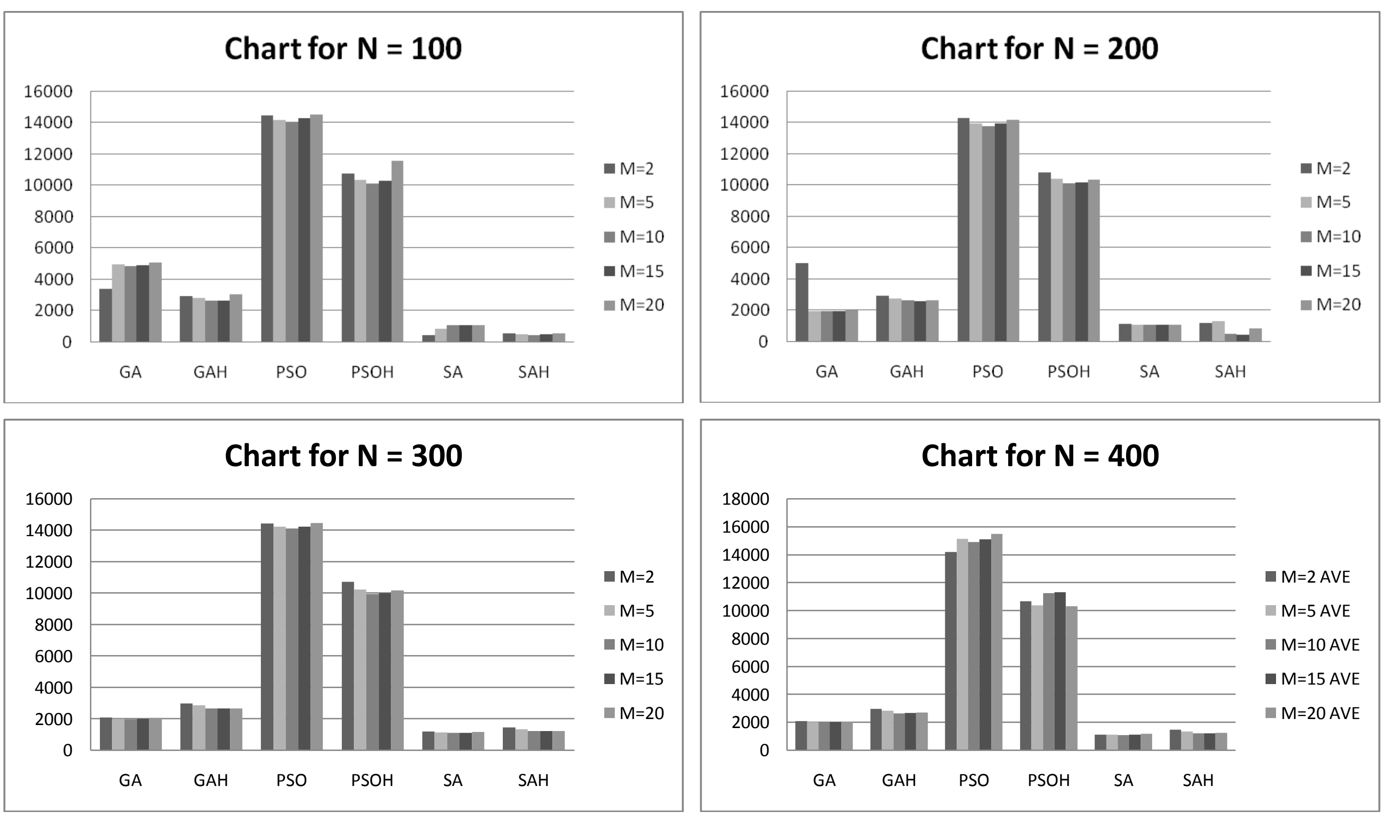

Figure 2. Performance time of the meta-heuristics.

Table 3. Homogeneous subsets using Scheffe's method (Harmonic mean sample size = 20.000).

Heuristics	N	Penalty: Subset for alpha = 0.05 1	2	3	4
SAH	20	116.7090			
GAH	20	133.2820	133.2820		
SA	20	163.1620	163.1620		
PSOH	20		240.9520	240.9520	
PSO	20			349.5210	
GA	20				494.1210
Sig.		0.865	0.082	0.077	1.000

Means for groups in homogeneous subsets are displayed.

Table 4. ANOVA.

Time	Sum of Squares	df	Mean square	F	Sig.
Between Groups	3.175E9	5	6.350E8	1634.682	0.000
Within Groups	4.429E7	114	388468.477		
Total	3.219E9	119			

Table 5. Post Hoc tests (multiple comparisons) using Scheffe's method.

(I) Heuristics	(J) Heuristics	Penalty: Mean difference (I-J)	Std. Error	Sig.	95% Confidence Interval: Lower bound	Upper bound
GA	GAH	360.83900*	33.93637	0.000	245.9076	475.7704
	PSO	144.60000*	33.93637	0.004	29.6686	259.5314
	PSOH	253.16900*	33.93637	0.000	138.2376	368.1004
	SA	330.95900*	33.93637	0.000	216.0276	445.8904
	SAH	377.41200*	33.93637	0.000	262.4806	492.3434
GAH	GA	-360.83900*	33.93637	0.000	-475.7704	-245.9076
	PSO	-216.23900*	33.93637	0.000	-331.1704	-101.3076
	PSOH	-107.67000	33.93637	0.082	-222.6014	7.2614
	SA	-29.88000	33.93637	0.978	-144.8114	85.0514
	SAH	16.57300	33.93637	0.999	-98.3584	131.5044
PSO	GA	-144.60000*	33.93637	0.004	-259.5314	-29.6686
	GAH	216.23900*	33.93637	0.000	101.3076	331.1704
	PSOH	108.56900	33.93637	0.077	-6.3624	223.5004
	SA	186.35900*	33.93637	0.000	71.4276	301.2904
	SAH	232.81200*	33.93637	0.000	117.8806	347.7434
PSOH	GA	-253.16900*	33.93637	0.000	-368.1004	-138.2376
	GAH	107.67000	33.93637	0.082	-7.2614	222.6014
	PSO	-108.56900	33.93637	0.077	-223.5004	6.3624
	SA	77.79000	33.93637	0.392	-37.1414	192.7214
	SAH	124.24300*	33.93637	0.025	9.3116	239.1744
SA	GA	-330.95900*	33.93637	0.000	-445.8904	-216.0276
	GAH	29.88000	33.93637	0.978	-85.0514	144.8114
	PSO	-186.35900*	33.93637	0.000	-301.2904	-71.4276
	PSOH	-77.79000	33.93637	0.392	-192.7214	37.1414
	SAH	46.45300	33.93637	0.865	-68.4784	161.3844

Table 5. Contd.

	GA	-377.41200*	33.93637	0.000	-492.3434	-262.4806
	GAH	-16.57300	33.93637	0.999	-131.5044	98.3584
SAH	PSO	-232.81200*	33.93637	0.000	-347.7434	-117.8806
	PSOH	-124.24300*	33.93637	0.025	-239.1744	-9.3116
	SA	-46.45300	33.93637	0.865	-161.3844	68.4784

* The mean difference is significant at the 0.05 level.

uniform and unrelated machines.

REFERENCES

Adamu M, Abass O (2010). Parallel machine scheduling to maximize the weighted number of just-in-time jobs. J. Appl. Sci. Technol. 15(1&2):27–34.

Adamu M, Adewunmi A (2012a). Metaheuristics for Scheduling on Parallel Machines to minimize the Weighted Number of Early and Tardy Jobs. Int. J. Phys. Sci. 7(10):1641-1652.

Adamu M, Adewunmi A (2012c). Single Machine Review to Minimize Weighted Number of Tardy Jobs. J. Ind. Manag. Optim. Submitted for publication.

Adamu MO, Adewunmi A (2012b). Minimizing the Weighted Number of Tardy Jobs on Multiple Machines: A Review. Asian J. Oper. Res.

Baptiste P, Jouglet A, Pape CL, Nuijten W (2000). A Constraint Based Approach to Minimize the Weighted Number of Late Jobs on Parallel Machines. Technical Report 2000/228, UMR, CNRS 6599, Heudiasyc, France.

Čepek O, Sung SC (2005). A Quadratic Time Algorithm to Maximize the Number of Just-In-Time Jobs on Identical Parallel Machines. Comput. Oper. Res. 32:3265-3271.

Chen Z, Powel WB (1999). Solving Parallel Machine Scheduling Problems by Column Generation. INFORMS J. Comput. 11(1):78-94.

Dauzère-Pérès S, Sevaux M (1999). Using Lagrangean Relation to Minimize the (Weighted) Number of Late Jobs on a Single Machine. National Contribution IFORS 1999, Beijing, P.R. of China (Technical Report 99/8 Ecole des Minesdes Nantes, France).

Garey MR, Johnson DS (1979). Computers and Intractability, A Guide to the Theory of NP Completeness. Freeman, San Francisco.

Graham RL, Lawler EL, Lenstra TK, Rinnooy Kan AHG (1979). Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey. Ann. Discrete Math. 5:287-326.

Hiraishi K, Levner E, Vlach M (2002). Scheduling of Parallel Identical Machines to Maximize the Weighted Number of Just-In-Time Jobs. Comput. Oper. Res. 29:841-848.

Ho JC, Chang YL (1995). Minimizing the Number of Tardy Jobs for *m* Parallel Machines. Eur. J. Oper. Res. 84:343-355.

Janiak A, Janiak WA, Januszkiewicz R (2009). Algorithms for Parallel Processor Scheduling with Distinct Due Windows and Unit-Time Jobs. Bull. Pol. Acad. Sci. Technol. Sci. 57(3):209-215.

Lann A, Mosheiov G (2003). A Note on the Maximum Number of On-Time Jobs on Parallel Identical Machines. Comput. Oper. Res. 30:1745-1749.

Li CL (1995). A Heuristic for Parallel Machine Scheduling with Agreeable Due Dates to Minimize the Number of Late Jobs. Comput. Oper. Res. 22(3):277-283.

Liu M, Wu C (2003). Scheduling Algorithm based on Evolutionary Computing in Identical Parallel Machine Production Line. Robot. Comput. Integr. Manuf. 19:401-407.

M'Hallah R, Bulfin RL (2005). Minimizing the Weighted Number of Tardy Jobs on Parallel Processors. Eur. J. Oper. Res. 160:471-484.

Sevaux M, Sörensen K (2005). VNS/TS for a Parallel Machine Scheduling Problem. MEC-VNS: 18th Mini Euro Conference pm VNS.

Sevaux M, Thomin P (2001). Heuristics and Metaheuristics for a parallel Machine Scheduling Problem: A Computational Evaluation. Proceedings of 4th Metaheuristics Int. Conf. pp. 411-415.

Süer GA (1997). Minimizing the Number of Tardy Jobs in Multi-Period Cell Loading Problems. Comput. Ind. Eng. 33(3&4):721-724.

Süer GA, Czajkiewicz Z, Baez E (1993). Minimizing the Number of Tardy Jobs in Identical Machine Scheduling. Proceedings of the 15th Conference on Computers and Industrial Engineering, Cocoa Beach, Florida.

Süer GA, Pico F, Santiago A (1997). Identical Machine Scheduling to Minimize the Number of Tardy Jobs when Lost-Splitting is Allowed. Comput. Ind. Eng. 33(1&2):271-280.

Sung SC, Vlach M (2001). Just-In-Time Scheduling on Parallel Machines. The European Operational Research Conference, Rotterdam, Netherlands.

Van Den Akker JM, Hoogeveen JA, Van De Velde SL (1999). Parallel Machine Scheduling by Column Generation. Oper. Res. 47(6):862-872.