International Journal of Physical Sciences Vol. 7(10), pp. 1641 - 1652, 2 March, 2012
Available online at http://www.academicjournals.org/IJPS
DOI: 10.5897/IJPS11.1741
ISSN 1992 - 1950 ©2012 Academic Journals

*Full Length Research Paper*

# Metaheuristics for scheduling on parallel machine to minimize weighted number of early and tardy jobs

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

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

Accepted 26 January, 2012

**This paper considers the scheduling of *n* jobs on *m* parallel machines to minimize the weighted number of early and tardy jobs. The single machine case of this problem has been shown to be NP- complete in the strong sense. This problem on *m* parallel machine is also NP complete in the strong sense and finding an optimal solution appears unlikely. The problem is formulated as an integer linear programming model. In this paper, we propose some meta-heuristics for solving this problem. Extensive computational experiments were performed which gave promising results.**

**Key words:** Scheduling, parallel machine, heuristics, metaheuristics, particle swarm optimization, genetic algorithm, simulated annealing, hybrid.

## INTRODUCTION

Scheduling jobs on the parallel machine to minimize the weighted number of tardy jobs is one of the well known problems in combinatorial optimization. In the last four decades, many results have appeared in the scheduling literatures that consider both earliness and tardiness penalties (Chen and Powell, 1999; Baptiste et al., 2000; Čepek and Sung, 2005). This problem is known to be NP-hard in the strong sense in the general case (Hiraishi et al., 2002). There is no efficient solution algorithm found yet for solving to optimality in polynomial time. This has led to recent interest in using meta-heuristic algorithms to address it.

In this work, we are considering minimizing the weighted number of early and tardy jobs on identical parallel machines. The problem has practical applications in real-world including in production and manufacturing industries, rental agencies, among others. Rental agencies (hotels, car rentals) must plan their reservations schedule to meet exactly the time requests of all customers. Customers require a room (car) within specific dates. If those dates cannot be met, the customers look for an alternative accommodation/agency implying loss of income for the agency. Alternatively, the agency can offer them a deal whereby; they will be scheduled on a different date, with a significant (fixed) compensation paid by the agency. The objective here is to maximize the number of customers scheduled as requested.

This paper proposes some metaheuristics including hybrids for solving the parallel machine scheduling problem. We explore solutions using Genetic algorithm (GA), Particle Swarm Optimization (PSO) and Simulated Annealing (SA) with promising results for large instance. These are further hybridized with four heuristics reported in Adamu and Abass (2010).

## LITERATURE REVIEW

Researchers have considered and developed various heuristics for various aspects of machine scheduling (Süer et al., 1993; Sevaux and Thomin, 2001; Adamu and Abass, 2010). Ho and Chang (1995) focused on two fundamental approaches namely, job-focused and machine-focused, to minimize the number of tardy jobs on the parallel processors. Three heuristics were proposed (two heuristics for the job focused approach and one for the machine-focused heuristic) for solving the problem of $P||\sum U_j$. Süer et al. (1993) proposed an integer

*Corresponding author. E-mail: adewumia@ukzn.ac.za.

programming formulation for the same problem aside three heuristics procedures for solving it. Süer (1997) considered the objective of minimizing the number of tardy jobs in a multi period environment. The results of his approaches were illustrated with a simple example. Süer et al. (1997) also addressed the problem of minimizing the number of late jobs on identical machine. They provided the mathematical modeling formulation and the result of simulation experiment. Van der Akker et al. (1999) solved the $P||\sum w_jC_j$ problem and explained how this can be extended to solve the identical and non-identical parallel machine weighted number of tardy jobs problems. They formulated the scheduling problem as a set covering problem with an exponential number of binary variables, *n* covering constraints, and a single side constraint. The LP relaxation of this formulation was then solved by column generation using a pseudo-polynomial algorithm to generate columns with negative reduced costs. Chen and Powell (1999) considered the problems $P||\sum w_jU_j$, $Q||\sum w_jU_j$ and $R||\sum w_jU_j$. The problem was formulated as an integer programming problem and Dantzig-Wolfe decomposition was used to reformulate it as a set partitioning problem. The lower bounds found by solving the linear programming relaxation through column generation were used to construct a branch and bound. Their results gave an average time to solve 10 machine 100 jobs instances as 0.33 h for $P||\sum w_jU_j$, 1.59 h for $Q||\sum w_jU_j$, and 0.91 h for $R||\sum w_jU_j$. Liu and Wu (2003) used evolutionary programming to solve the unweighted case of the problem. They obtained a better result when compared to that found in Süer et al. (1993). Furthermore, M'Hallah and Bulfin (2005) considered the three problems as defined by Chen and Powell (1999) to minimize the weighted number of tardy jobs. They provided a branch and bound algorithm that used bounds from a surrogate relaxation resulting in a multiple-choice knapsack. They conducted computational experiments that indicate problems with 400 jobs and several machines can be solved quickly. They obtained a wide timing gap when compared with that of Chen and Powell (1999). The longest branch and bound time in any of their sets was 5.17 CPU seconds.

Similarly, Sevaux and Thomin (2001) addressed the NP-hard problem to minimize the weighted number of late jobs with release time. 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 found in 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 (2002) 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 found in 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 heuristic algorithm is presented and a dynamic programming lower bounding procedure developed.

For the case of due window considered in this paper, Hiraishi et al. (2002) addressed the non-preemptive scheduling of n jobs that are completed exactly at their due dates. They showed that the 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 Hiraishi et al. (2002) is solvable in polynomial time (exponential in the number of machines) regardless of 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 defined by Hiraishi et al. (2002) with great improvement in time complexity. Čepek and Sung (2005) considered the same problem, gave a corrected form of the greedy algorithm in Lann and Mosheiov (2003) and presented a new quadratic time algorithm that solves 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 was performed. Janiak et al. (2009) 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 $O(n^3)$ complexity $(Pm|p_j = 1, r_j$, agreeable ET weights$|\sum w_j(U_j + V_j))$.

## PROBLEM FORMULATION

For *n* independent jobs, scheduled on m machines, which are simultaneously available from time zero, each having an interval rather than a point in time, called *due window* of the job, and the left end and the right end of the window are called, respectively, the *earliest due date* $a_j \geq 0$ (that is, the instant at which a job becomes available for delivery), and the *latest due date* $d_j \geq 0$ (an instant by which processing or delivery of a job must be completed). There is no penalty when a job is completed within the job's due window, but earliness (tardiness) penalty is incurred if a job is completed before the job's earliest due date (after the job's latest due date).

For any given schedule S, let $p_j$, $t_{ij}$ and $C_{ij}(S) = t_{ij} + p_j$ represent the processing time, 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 $a_j \leq t_{ij} + p_j \leq d_j$. Furthermore, define $U_j$ and $V_j$ as follows:

$$U_j = \begin{cases} 1, & if\ C_{ij}(S) < a_j \\ 0, & otherwise \end{cases} \qquad (1)$$

$$V_j = \begin{cases} 1, & if\ C_{ij}(S) > d_j \\ 0, & otherwise \end{cases} \qquad (2)$$

Let $w_j \geq 0$ be the weights for scheduling job *j* ($j \in N$) early and tardy. The objective is to find the feasible schedule S* which minimizes the weighted number of early and tardy jobs on identical parallel machine given as

$$P:\ min \sum_{i=1}^{m} \sum_{j=1}^{n} w_j (U_j + V_j) \qquad (3)$$

Taking the dual of this objective function, let $x_{ij}$ be one if job *j* is on-time and zero otherwise and $C_{ik}$ the completion time of the last job on machine *i* before job *j*. Then

$$P: \{min \sum_{i=1}^{m} \sum_{j=1}^{n} w_j (U_j + V_j)\}\ max \sum_{i=1}^{m} \sum_{j=1}^{n} w_j x_{ij} \qquad (4)$$

Such that

$$a_j \leq \{\max_{k=1}^{j} \{C_{ik-1}(S), a_j - p_j\} + p_j\} x_{ij}\ i=1,...,m;\ j = 1, . . . , n, \qquad (5)$$

$$\{\max_{k=1}^{j} \{C_{ik-1}(S), a_j - p_j\} + p_j\} x_{ij} \leq d_j\ i=1,...,m;\ j = 1, . . . , n, \qquad (6)$$

$\sum_{i=1,m} x_{ij} \leq 1$ j = 1, . . . , (7)

$$x_{ij} \in \{0,1\}\ i = 1, . . . ,m ;\ j = 1, . . . , n \qquad (8)$$

Constraint (5) ensures job j is not finished before the job's earliest due date $a_j$. Constraint (6) is the completion time of job j is not greater than the latest due date $d_j$. Constraint (7) ensures that a job is assigned to at most one machine, and Constraint (8) forces a job to be either on-time or early/tardy; one if on-time and zero otherwise.

## HEURISTICS AND META-HEURISTICS

### Greedy heuristics

Adamu and Abass (2010) have proposed four greedy heuristics which attempt to provide near optimal solutions to the parallel machine scheduling problem. The first heuristic (W01) sorts jobs (tasks) in ascending order according to earliest due date (that is, the earliest time in which the task may be finished). If two jobs have the same earliest due date, then the tie is broken by placing the job with the highest processing time first. Jobs are then assigned onto machines using this sorted order. The second heuristic (WO2) is identical to the first except for using a different tie-breaking rule. Instead of using the highest processing time, the highest weighted processing time is used (that is, weight / processing time). The third heuristic (DO1) is the same as the first except that instead of sorting by earliest due date, jobs are sorted by latest due date (that is, latest due time -processing time). The final heuristic (DO2) combines the sorting method used in the third heuristic with the tie-breaking rule used in the second heuristic.

Though the results of these greedy heuristics are promising, this paper further investigates them and explores their hybridization with some meta-heuristics in search of better results.

### Genetic algorithm

Genetic algorithm (GA) (Goldberg, 1989) has been proved successfully as a meta-heuristics for solving optimization problems. Based on the biological inspiration of evolution, survival of the fittest, crossover and mutation, this paper investigates GA's performance for the problem defined previously under Problem Formulation for comparative study with that of the greedy heuristics.

We adopted a string data structure for our problem. Each job is fixed to a gene in the chromosome implying that the chromosome has length n (where n is the number of jobs). Each gene also has 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). The GA operators were designed to influence both the machine number and the order. The fitness evaluation function was designed to calculate 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 (with priority based 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). The fitness range is such that a lower value implies better performance.

In our simulation experiment, we used the single-point crossover operator 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 experiments with variations of these operators were performed. The pseudocode of the GA used is presented in Figure 1.

### Particle swarm optimization

Solution with PSO for the parallel machine scheduling problem was investigated in this work. PSO is a population based technique inspired by the flocking behavior of birds that is influenced by both a particle's best position as well as the global best position in the overall population (Parsopoulos and Vrahatis, 2010). We explore the possibility of PSO for obtaining local optima which can then drive the search for the global optimum solution. The PSO algorithm requires a solution representation or encoding such that each particle is an instance 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. Our representation is as follows: Each particle contains a number in [0,m), where m is the number of machine. This number represents the machine on which the particle is scheduled and is simply truncated to convert to the discrete space. For the order of scheduling, each particle contains a number in [0,1) which represents the order of scheduling relative to other particles on the same machine. A job with lower number will be scheduled before the jobs with higher numbers. Finally, a method was devised to convert the encoding into a valid schedule. This is done 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

```
Generate a population of randomly initialized individuals.
iterations ← 0
repeat
  for i = 1 → popSize do
    Perform crosssover with probability crossoverRate.
  end for
  for i = 1 → popSize do
    for j = 1 → numJobs do
      Mutate machine with probability mutationRate.
    end for
  end for
  for i = 1 → popSize do
    for j = 1 → numJobs do
      Mutate order with probability mutationRate.
  end for
    end for
  Use selection to form a new population of individuals.
  iterations ← iterations + 1
  until iterations ≥ numIterations
  Return the fitness of the best individual.
```

**Figure 1.** Pseudo code of GA for parallel machine scheduling.

solution (which should ideally be as small as possible). The pseudo code of the PSO used is presented in Figure 2.

## Simulated annealing

We further explore the single-solution SA metaheuristics (Kirkpatrick et al., 1983) for the parallel machine scheduling problem. Based on the principle of annealing in thermodynamics, the exploration of single solution in SA requires fewer computations in comparison with the population-based technique. We thus expect the execution times for this technique to be quicker. Moreover, we hope SA, in all likelihood, will 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). For the SA algorithm, we use a similar representation as that in GA The fitness evaluation is also done in similar manner. For the neighbourhood selection, we allow an element to be 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 neighbour, we ensure that good exploration is achieved while seeking to escape from local optimum solutions.

## Improvements and hybrids

Aside the fundamental implementation of the basic heuristics and meta-heuristics mentioned previously, a number of improvements were experimented in search of better solution.

Aside the original GA which was tested using 1-point crossover, random mutation for machines, swap mutation for order and tournament selection, we tried 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 $i = 1 \rightarrow PopSize$ do
            Construct particle with randomly initialized machine number and order.
        end for
        repeat
            $pbest \leftarrow 2^{31}$
            for $i = 1 \rightarrow PopSize$ do
                $fitness \leftarrow calcfitness(pop[i])$
                if $fitness < pbest$ then
                    $pbest \leftarrow fitness$
                end if
                if $fitness < fitness(gbest)$ then
                    gbest ← pop[i]
                end if
            end for
            for $i = 1 \rightarrow PopSize$ do
                $v[i + 1] \leftarrow wv[i] + r_1c_1(pbest - pos[i]) + r_2c_2(gbest - pos[i])$
                $pos[i + 1] \leftarrow pos[i] + v[i]$ (ensuring to clamp the position within range)
            end for
            $iterations \leftarrow iterations + 1$
        until $iterations \geq numIterations$
```

**Figure 2.** Pseudo code of PSO for parallel machine scheduling.

(for order) were explored. User can decide on a combination of operators to use.

The greedy heuristics reported in Adamu and Abass (2010) were incorporated as local search for the GA, PSO and SA meta-heuristics. Adamu and Abass (2010) reported that the performance of the greedy heuristics hinge on the order of assigning jobs to machines. We explored the same ordering mechanisms as in WO1, WO2, DO1, DO2 (Adamu and Abass, 2010) and incorporated them into GA, PSO, and SA to form hybrids. In our overall experiment, any of the four greedy heuristics can be hybridized with GA, PSO or SA. In this paper however, we only reported four new hybrids namely, *GAHybrid, PSOHybrid, PSOGA and SAHybrid*. Past research (Premalatha and Natarajan, 2009) has shown that PSO can become stuck in local minimum and has proposed the combination of PSO with some GA features to prevent premature convergence. Premalatha and Natarajan (2009) suggested the incorporation of mutation operator if particle's local best remain unchanged over a period of time. This motivated the PSOGA hybrid for this research work. The PSOGA hybrid allows any pBest position which was not updated (that is, the particle did not find a new best) to be mutated by applying a random mutation to the machine number and swap mutation to the order.

## COMPUTATIONAL ANALYSIS

Simulation experiments were carried out using an open-source eclipse® integrated development environment (IDE) for Java developer, release 3.3 on Microsoft Windows XP platform running on a Pentium dual 1.86 GHz, 782 MHz, and 2 GB of Ram The system clock was used to track execution time in milliseconds.

### Data generation

The heuristics were tested on problems generated with 500, 1000, 1500, 2000 and 2500 jobs similar to that used in previous studies (Adamu and Abass, 2010; Baptiste et al., 2000; Ho and Chang, 1995; 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* representing 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, (5000/*n*)*k1), and the integer latest due date ($d_j$) was randomly generated in the interval ($a_j + p_j$, $a_j + p_j$ + (5000/*n*)*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.

### Parameters

In the simulation experiment, parameters were combined to control each meta-heuristic to determine an optimal parameter set for the scheduling problem. This led to a set of parameter combination used in obtaining the final results. For the GA and its hybrids, the population size was set to 10, random mutation (for machines) with a mutation rate of 0.01, swap mutation (for order) with a rate of 0.01, uniform crossover at a rate of 0.5, tournament selection with a *k* value set at 40% of the population size and the number of iterations set at 2000.

The optimal parameters combination for PSO are population size of 50, a momentum value (w) of 0.3, c1 of 2, c2 of 2 and the number of iterations set at 2000. Similarly, optimal parameter combination for SA is initial temperature of 25, a final temperature of 0.01 and a geometrical decreasing factor (β) of 0.999.

## RESULTS AND DISCUSSION

The results for GA, PSO, SA, WO1, WO2, DO1, DO2, GAHybrid, PSOHybrid, PSOGA and SAHybrid were analyzed and presented in Appendix. In Appendix Tables 1 to 5, each cell consists of two numbers. The first number is the average weight of the schedule produced over 50 runs. The number in parenthesis is the average time in milliseconds that the algorithm takes to complete. The times for the greedy heuristics are not included since the time is very small (averagely 0.115 µs). Also, the graphs in Appendix Figures 3 to 8 show results obtained with the algorithms for each value of n in [500, 2500]. The graphs compared the relative performance (penalty) of each of the 11 algorithms compared to the number of machines used. Two graphs were used to show the computational times of the meta-heuristics when N = 1000 and N = 2000.

Generally from these results, it is very clear that majority of the meta-heuristics outperformed the greedy heuristics. However, the greedy heuristics achieved better results than GA in all of the categories except m is 2 for all N jobs. The results showed it to be in the region of 16000 times slower than the greedy heuristics on the overall. GA hybridized with DO2 (GAH) achieved better results than traditional GA for all of the test cases although the hybrid took an average of about 50% longer to run than GA (averagely 2 to 3 s) (Appendix). GAH performed better than WO1, WO2, DO1, and DO2 heuristics except when the number of machines was high (that is, 20) and the number of jobs was low (e.g. 500, 1000 and 1500).

PSO and its hybrid (PSOH) produced lower weight than all the greedy heuristics except when the number of machines was generally small (that is, 2) (Furthermore, they were far slower than the greedy heuristics (over 100000 times slower for PSO and 85000 for PSOH). This is understandable since PSO is a population-based algorithm with lots of parallel computations at each step. Hybridizing PSO with the DO2 (PSOH) produced results which were worst than PSO for almost all case except when N = 500. PSOH was about 1.2 times faster than PSO. Meanwhile, the PSO outperformed GA in several cases considered.

SA produced far better results than those of the greedy heuristics and in fact other meta-heuristics (Appendix Tables 1 to 5). On the average, SA produced a weight of about 100 less than what the best greedy heuristic produces (Appendix Tables 1 to 5). The only setback when compared with the greedy heuristic is that it was slower (over 2000 times slower). However, it was more than 4 times faster than the PSO with superior results (Appendix Tables 1 to 5). Hybridizing SA with the DO2 greedy heuristic (SAH) produced results that were slightly better than the SA solutions for a large number of machines except majorly for number of machines equal to two (Appendix: Table 1). It is, however, about 1.3 times slower than the original SA algorithm.

Finally, the hybrid of PSO and GA (PSOGAH) had a similar performance with that of PSO in both penalty incurred and execution time (Appendix: Table 1). Thus, one can conclude that adding the mutation operator to the PSO algorithm had very little effect on its performance. On the average for all test cases, PSOGAH still achieved better results than the greedy heuristics except when machines are 15 and 20 in number.

## Conclusion

Adamu and Abass (2010) showed that greedy heuristics could be used effectively to solve the parallel machine scheduling problem. In this paper, we further investigated the use of GA, PSO and SA with some hybrids in search of better results. The three meta-heuristics (GA, PSO and SA) showed an improved average performance over all the greedy heuristics. Hybridizations of the meta-heuristics with the greedy heuristics did not produce much noticeable performance increase except in GA. In the overall, SA proved to be the best performing meta-heuristic with the lowest average weight (penalty) for each of the test cases and lowest execution time compared with other meta-heuristics.

There is little doubt that SA could be used to achieve near optimal solutions to the parallel machine scheduling problem in the real-world. However, considerations need to be made as to whether the run time of the algorithm (1 to 3 s) may affect the overall outcome. For example, the SA algorithm would be ideal for a large scale application such as scheduling tasks on a building site where wasting 2 to 3 s would not harm any overall outcomes. On the other hand, if the SA algorithm was to be used on a computer’s operating system for task scheduling, the 2 to 3 s wait would be extremely costly and it would instead make sense to use a speedy greedy heuristic such as DO2 (while taking a performance decrease). Further work would seek better improvement as well as exact solution to the problem.

## REFERENCES

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

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. Comp. Op Res., 32: 3265-3271.

Chen Z, Powel WB (1999). Solving parallel machine scheduling problems by column generation. INFORMS J. Comp., 11(1): 78-94.

Dauzère-Pérès S, Sevaux M (2002). Using Lagrangean relaxation to minimize the (weighted) number of late jobs on a single machine. Nat. Res. Log., 50(3).

Goldberg DE (1989), *Genetic Algorithms in Search, Optimization and Machine Learning,* Kluwer Academic Publishers, Boston, MA, p, 432.

Hiraishi K, Levner E, Vlach M (2002). Scheduling of parallel identical machines to maximize the weighted number of just-in-time jobs. Comp. Op. 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. Bulletin of the Polish Academy of Sci. Tech. Sci., 57(3): 209-215.

Kirkpatrick S, Gelatt CD, Vecchi MP (1983). Optimization by simulated annealing. *Science,* 220 (4598): 671–680.

Lann A, Mosheiov G (2003). A note on the maximum number of on-time jobs on parallel identical machines. Comp. 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. Comp. Oper. Res., 22(3): 277-283.

Liu M, Wu C (2003). Scheduling algorithm based on evolutionary computing in identical parallel machine production line. Rob. Comp. Int. 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.

Parsopoulos KE, Vrahatis MN (2010). *Particle Swarm Optimization and Intelligence: Advances and Applications*. IGI Global, USA. ISBN: 978–1–61520–666–7.

Premalatha K, Natarajan A (2009). Hybrid PSO and GA for global maximization. Int. J. Op. Prob. Comp. Sci. Math., 2(4): 597-608.

Sevaux M, Sörensen K (2005). VNS/TS for a parallel machine scheduling problem. MEC-VNS: 18th Mini Euro Conference on VNS.

Sevaux M, Thomin P (2001). Heuristics and metaheuristics for a parallel machine scheduling problem: A computational evaluation. Proceedings of 4th International Conference on Metaheuristics, pp. 411-415.

Süer GA (1997). Minimizing the number of tardy jobs in multi-period cell loading problems. Comp. Ind. Eng., 33(3-4): 721-724.

Süer GA, Czajkiewicz Z, Baez E (1993). Minimizing the number of tardy jobs in identical machine scheduling. Comp. Ind. Eng., 25 (1-4): 243-246.

Süer GA, Pico F, Santiago A (1997). Identical machine scheduling to minimize the number of tardy jobs when lost-splitting is allowed. Comp. Ind. Eng., 33(1-2): 271-280.

Sung SC, Vlach M (2001). Just-in-time scheduling on parallel machines. Proceedings of the European Operational Research conference held in 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.

## APPENDIX

**Table 1.** Performance of heuristics at N = 500.

| Job | Number of machines / Heuristics: weight (time) | 2 | 5 | 10 | 15 | 20 |
|---|---|---|---|---|---|---|
| 500 | WO1 | 795.40(-) | 683.04(-) | 516.80(-) | 374.30(-) | 245.38(-) |
| | WO2 | 795.78(-) | 685.58(-) | 520.54(-) | 375.36(-) | 247.12(-) |
| | DO1 | 804.46(-) | 697.90(-) | 515.06(-) | 314.90(-) | 90.44(-) |
| | DO2 | 804.30(-) | 695.46 (-) | 516.16 (-) | 319.68 (-) | 89.96(-) |
| | GA | 788.62 (1958.50) | 713.94(1913.78) | 618.72(1852.18) | 549.26(1884.72) | 487.66(1941.18) |
| | GAH | 727.38(2827.48) | 553.88(2714.96) | 390.10(2565.58) | 287.48(2557.42) | 207.24(2582.84) |
| | PSO | 698.02(15757.86) | 625.76(13398.50) | 543.84(13227.90) | 486.10(14378.12) | 439.26(14193.48) |
| | PSOH | 708.68(10298.82) | 636.78(9932.50) | 545.14(9635.26) | 471.44(9609.66) | 415.00(9787.86) |
| | SA | 694.58(271.20) | 556.34(256.26) | 402.90(248.14) | 295.16(255.34) | 220.02(248.14) |
| | SAH | 743.66(344.26) | 552.72(309.86) | 366.26(438.42) | 251.48(664.40) | 171.98(690.96) |
| | PSO-GAH | 709.72(13626.16) | 641.38(13349.94) | 546.12(13170.00) | 487.04(13382.52) | 447.80(13785.58) |

**Table 2.** Performance of Heuristics at N = 1000

| Job | Number of machines / Heuristics: weight (time) | 2 | 5 | 10 | 15 | 20 |
|---|---|---|---|---|---|---|
| 1000 | WO1 | 826.58(-) | 754.18(-) | 639.00(-) | 531.22(-) | 434.94(-) |
| | WO2 | 829.34(-) | 758.58(-) | 640.86(-) | 536.50(-) | 436.52(-) |
| | DO1 | 835.64(-) | 762.54(-) | 634.12(-) | 494.96(-) | 340.82(-) |
| | DO2 | 836.10(-) | 762.34(-) | 635.10(-) | 494.60(-) | 337.48(-) |
| | GA | 822.24(1947.78) | 770.60 (1884.76) | 695.70(1869.28) | 631.14(1880.96) | 586.18(1925.86) |
| | GAH | 785.82(2838.18) | 660.16(2721.26) | 528.22(2585.26) | 429.96(2568.76) | 357.32(2601.44) |
| | PSO | 757.00(13446.50) | 705.10(13272.42) | 638.74(13288.40) | 585.24(13504.34) | 544.26(14722.22) |
| | PSOH | 778.12(10281.94) | 727.48(9850.92) | 658.22(9648.12) | 604.48(9663.82) | 550.10(10050.04) |
| | SA | 763.12(269.36) | 650.10(254.76) | 524.02(254.96) | 433.12(251.86) | 357.96(265.08) |
| | SAH | 796.62(779.76) | 664.28(713.16) | 510.38(665.66) | 406.48(658.48) | 327.16(676.86) |
| | PSO-GAH | 763.08(13662.58) | 717.12(13331.86) | 640.94(13228.84) | 589.72(13422.38) | 549.56(13779.68) |

**Table 3.** Performance of heuristics at N = 1500.

| Job | Number of machines / Heuristics: weight (time) | 2 | 5 | 10 | 15 | 20 |
|---|---|---|---|---|---|---|
| 1500 | WO1 | 832.74(-) | 770.38(-) | 682.24(-) | 596.36(-) | 514.22(-) |
| | WO2 | 833.54(-) | 777.80(-) | 685.38(-) | 597.02(-) | 518.50(-) |
| | DO1 | 835.92(-) | 773.88(-) | 663.82(-) | 547.84(-) | 423.98(-) |
| | DO2 | 836.36(-) | 774.42(-) | 664.26(-) | 548.90(-) | 421.24(-) |
| | GA | 825.46(1956.92) | 784.86 (1881.84) | 720.70(1854.10) | 672.06(1884.00) | 629.20(1932.56) |
| | GAH | 798.32(2854..46) | 702.94 (2738.16) | 576.36(2582.48) | 494.32(2593.44) | 428.14(2638.40) |
| | PSO | 771.9(13981.86) | 729.60 (14621.54) | 665.90(17273.08) | 627.66(18210.82) | 591.30(18686.38) |
| | PSOH | 791.74(10406.98) | 749.30 (9885.60) | 695.40(9693.12) | 646.58(9815.16) | 608.48(9970.96) |
| | SA | 775.98(268.08) | 685.58 (254.36) | 575.36(257.22) | 491.82(261.90) | 424.28(268.22) |
| | SAH | 803.54(797.76) | 699.36 (731.84) | 569.88(676.20) | 477. 78(669.78) | 403.84 (675.04) |
| | PSO-GAH | 773.3(13619.38) | 737.24 (13260.28) | 679.38(13177.28) | 628.66(13369.06) | 592.54(13754.38) |

**Table 4.** Performance of Heuristics at N = 2000

| Jobs | Number of machines / Heuristics: weight (time) | 2 | 5 | 10 | 15 | 20 |
|---|---|---|---|---|---|---|
| 2000 | WO1 | 829.54(-) | 778.78(-) | 700.12(-) | 623.34(-) | 546.74(-) |
| | WO2 | 831.10(-) | 784.10(-) | 701.66(-) | 625.12(-) | 549.28(-) |
| | DO1 | 831.22(-) | 777.86(-) | 688.18(-) | 580.76(-) | 462.50(-) |
| | DO2 | 831.58(-) | 778.66(-) | 688.56(-) | 581.26(-) | 464.00(-) |
| | GA | 823.40(1946.54) | 784.88(1881.54) | 730.46(1849.72) | 688.76(1879.64 | 643.24 (1934.02) |
| | GAH | 795.94(2867.00) | 710.38(2754.36) | 600.26(2612.18) | 520.98(2605.06) | 456.74(2650.28) |
| | PSO | 774.70(16126.26) | 731.18(17478.18) | 683.12(15811.24) | 635.48(15822.74) | 607.1413(635.64) |
| | PSOH | 793.14(10476.48) | 755.82(10010.02) | 707.60(10634.98) | 664.06(9746.00) | 624.82(9989.28) |
| | SA | 781.90(272.50) | 698.56(254.64) | 593.36 (247.84) | 515.36(252.22) | 447.76(263.78) |
| | SAH | 798.88(799.34) | 705.24(737.86) | 592.62(673.42) | 506.26(668.84) | 436.92(682.54) |
| | PSO-GAH | 773.66(13535.92) | 740.60(13419.74) | 685.40(13241.24) | 644.08(13564.98) | 608.58(13844.70) |

**Table 5.** Performance of Heuristics at N = 2500

| Jobs | Number of machines / Heuristics: weight (time) | 2 | 5 | 10 | 15 | 20 |
|---|---|---|---|---|---|---|
| 2500 | WO1 | 844.44(-) | 796.50(-) | 721.06(-) | 643.14(-) | 570.76(-) |
| | WO2 | 848.74(-) | 798.46(-) | 724.84(-) | 649.88(-) | 575.88(-) |
| | DO1 | 845.44(-) | 797.52(-) | 712.12(-) | 617.10(-) | 506.22(-) |
| | DO2 | 845.34(-) | 798.38(-) | 713.16(-) | 616.50(-) | 505.68(-) |
| | GA | 836.44(1943.10) | 802.54(1880.40) | 755.62(1858.68) | 713.12(1886.82) | 672.08(1925.92) |
| | GAH | 812.80(2911.78) | 736.84(2780.00) | 632.36(2621.30) | 556.42(2639.40) | 497.78(2665.02) |
| | PSO | 793.84(13425.94) | 749.94(13254.42) | 703.68(13208.10) | 665.74(13338.98) | 633.50(13641.84) |
| | PSOH | 813.40(10444.02) | 776.76(10054.72) | 728.60(9767.76) | 689.74(10715.24) | 656.34(9992.54) |
| | SA | 799.86(275.78) | 719.24 (364.12) | 624.30(584.68) | 549.04(597.78) | 487.60(618.42) |
| | SAH | 815.20(812.20) | 729.84(737.62) | 626.94(678.04) | 545.30(673.46) | 479.97(688.50) |
| | PSO-GAH | 791.64(13698.72) | 761.04(13342.50) | 709.70(13187.48) | 667.20(13357.50) | 633.06(13702.86 |

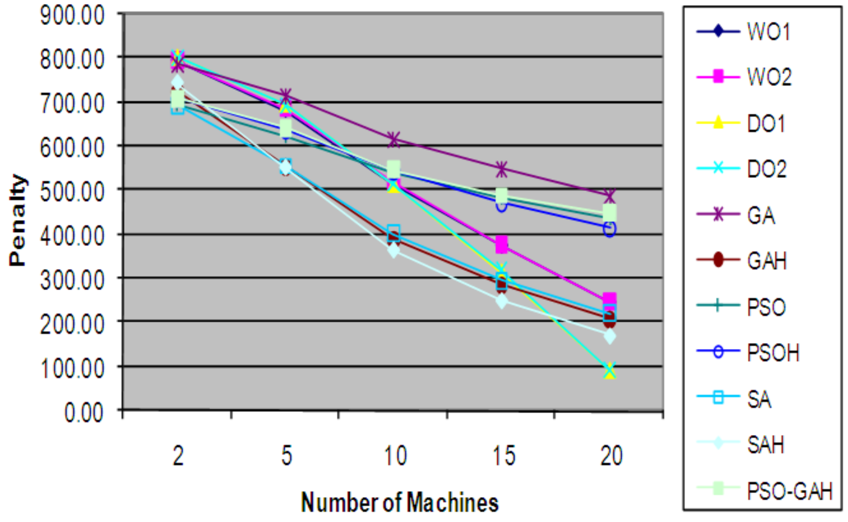

**Figure 3.** Comparison of heuristics and meta-hauristics for N = 500.

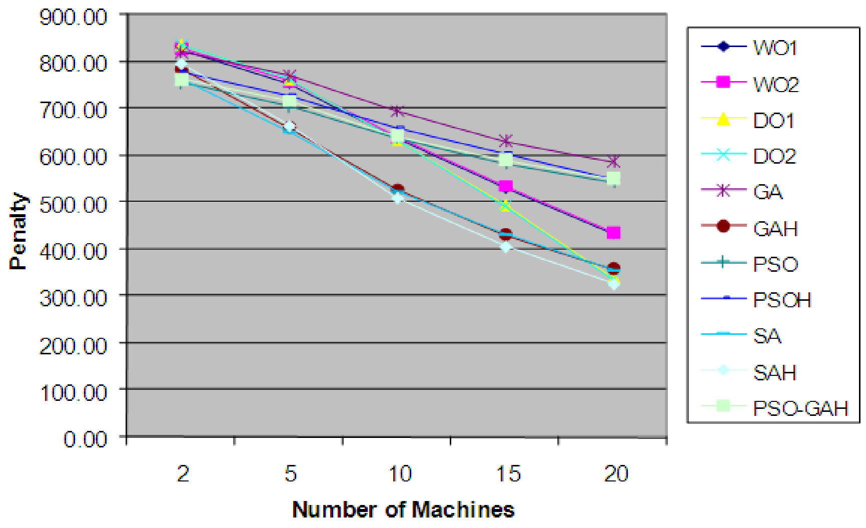

**Figure 4.** Comparison of heuristics and meta-heuristics for N = 1000.

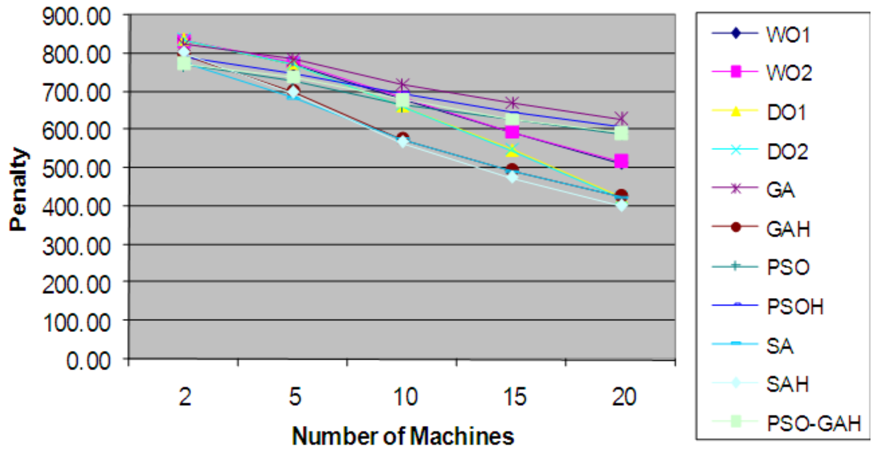

**Figure 5.** Comparison of heuristics and meta-heuristics for N = 1500.

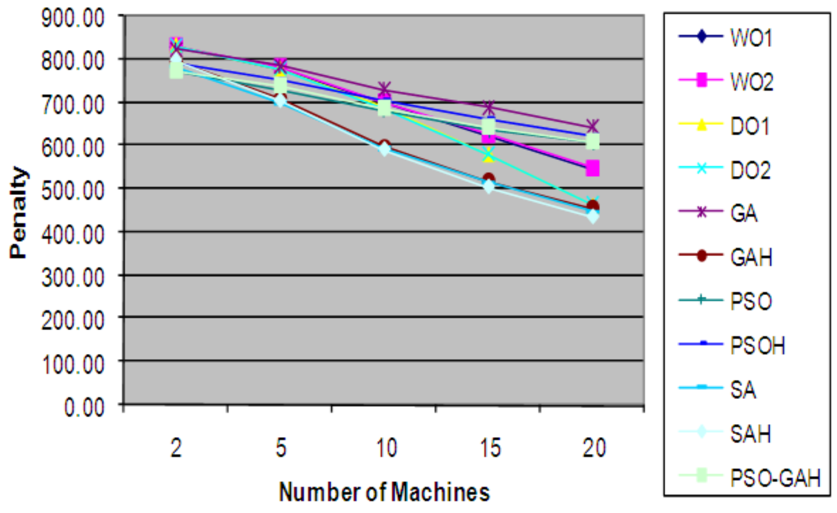

**Figure 6.** Comparison of heuristics and meta-heuristics for N = 2000.

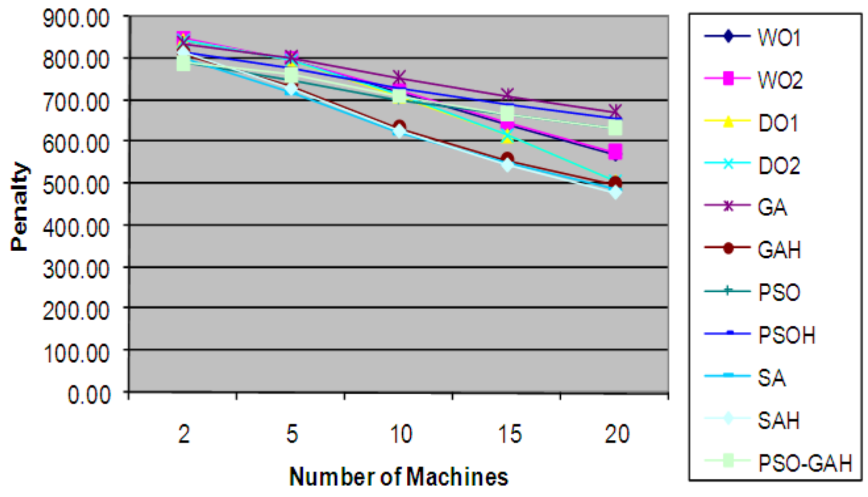

**Figure 7** Comparison of heuristics and meta-heuristics for N = 2500.

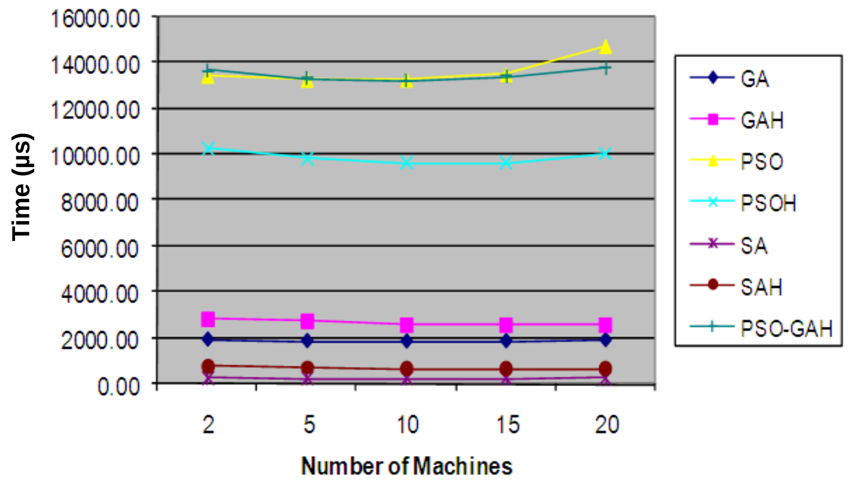

**Figure 8.** Comparison of meta-heuristics computational time for N = 1000.