UNWEIGHTED PARALLEL MACHINE SCHEDULING: A META-HEURISTIC APPROACH

M.O. Adamu
Department of Mathematics, University of Lagos, Lagos, Nigeria.
madamu@unilag.edu.ng

A. Adewunmi
School of Mathematics, Statistics and Computer Science, University of Kwazulu-Natal, Durban, South Africa.
adewumia@ukzn.ac.za

ABSTRACT

This article considers the due window scheduling problem to minimize the number of early and tardy jobs on identical parallel machines (Pm||Σ(U_j + V_j)). This problem is known to be NP complete and finding an optimal solution is unlikely. Three meta-heuristics and their hybrids are proposed with extensive computational experiments conducted. The overall best among them is Simulated Annealing hybrid. Detailed comparative tests were also conducted to analyze the different heuristics.

Keywords: Parallel Machine, Heuristics, Just-In-Time, Meta-heuristic, NP Complete

1. Introduction

Due to more emphasis on satisfying customers in service provision, due-date related objectives are becoming important in scheduling. In this article, the objective of minimizing the number of early and tardy jobs is considered; a situation where more jobs are completed between their due windows (due-dates). This objective is practical in real world situation to attain a better customer service rating. A comparative study of several heuristics is considered for solving this problem. In the identical parallel machine problem, *n* jobs are to be processed on *m* machines assuming the following facts:

- A job is completed when processed by any of the machines
- A machine can process only one job at a time
- Once a job is being processed, it cannot be interrupted
- All jobs are available from time zero
- The weights of the jobs are equal
- All processed jobs are completed within their due windows

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 (see Adamu and Adewunmi [1] and on single machine (see Adamu and Adewunmi [2])). Garey and Johnson [3] have shown our problem to be NP-complete and finding an optimal solution appears unlikely. Using the three-field notation of Graham et al [4], the problem is represented as Pm| |Σ(U_j + V_j). Scheduling to minimize the (weighted) number of tardy jobs have been considered by Ho and Chang [5], Süer *et al* [6], Süer [7], Süer *et al* [8], Van der Akker [9], Chen and Powell [10], Liu and Wu [11], and M'Hallah and Bulfin [12]. Other articles are Sevaux and Thomin [13], Baptiste *et al* [14], Dauzère-Pérès and Sevaux [15], Sevaux and Sörensen [16], Li [17], Hiraishi *et al* [18], Sung and Vlach [19], Lann and Mosheiov [20], Čepek and Sung [21] and Janiak *et al* [22]. Adamu and Abass [23] proposed four greedy heuristics for the Pm||Σw_j (U_j + V_j) problem and extensive computational experiments performed. Adamu and Adewunmi [24] proposed some Meta-heuristics and their hybrids to solve the problem considered by Adamu and Abass [23] and found them performing better.

2. Problem Formulation

Let ther be an independent set, N = {1,2, . . . , n} of jobs that are to be scheduled on *m* parallel identical machines, which are immediately 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 (i.e. the instant at which a job becomes available for delivery), and the latest due date (i.e. 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 $j \epsilon$ N has a processing time p_j, earliest due date a_j, and latest due date d_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 number of early and tardy jobs (Liu and Wu [11])

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

Where we define that

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

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

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

Therefore, the problem of scheduling to minimize the number of tardy jobs on identical parallel machines can be formulated as W.

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

Min

$$\sum_{i=1}^{m} \sum_{j=1}^{n} U_j =$$

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

3. Heuristic and Meta-heuristics

3.1 Greedy Heuristic

Adamu and Abass [23] 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 (i.e. latest due time - processing time) and ties broken by the highest weighted processing time is used (i.e. 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. Similar codes used in Adamu and Adewunmi [24] shall be used to solve the problem of minimizing the number of early and tardy jobs on parallel machines.

3.2 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.

1) 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.

2) *Algorithm:* Below is a basic pseudo code of the genetic algorithm which was used.

```
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.
```

3.3 Particle Swarm Optimization

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.

1) 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:

- 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.
- 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.

2) Algorithm: Below is a basic pseudo code of the PSO which was used.

```
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
            $gbest \leftarrow pop[i]$
        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 $iterations \geq numIterations$
```

3.4 *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 (i.e. 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).

1) 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.

2) Algorithm: This is the basic algorithm used in the SA technique:

```
Generate a randomly initialized solution sol.
repeat
    for $i \rightarrow 10$ do
        find a neighbor of sol and call it $solPrime$.
        if $fitness(solPrime) < fitness(sol)$ then
            $sol \leftarrow solPrime$
        else
            if $e^{fitness(sol) - fitness(solPrime)} > rand(0..1)$ then
                $sol \leftarrow solPrime$
            end if
        end if
    end for
    $temp \leftarrow temp*beta$
until $temp \leq templ$
```

4. Computational Analysis and Result

4.1 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[22], Ho and Chang [5], Baptiste et al [14] and M'Hallah and Bulfin [12]. 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, i.e., for each value of n, 160 instances were generated 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.

4.2 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 in the parameters section.

4.3 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 hybridisation 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 (e.g. swap mutation in Chromosome) was removed.

4.4 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.

5. Discussion on Results

In this section, the results of the algorithms are shown, including the hybridizations. In the four tables 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 (see Figure 1) 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.
It should be clear from both the Table 2 and the charts (Figures 1) that the Simulated Annealing Hybrid (SAH) out performed the other meta-heuristics in almost all points. Its average performance time is 0.5 seconds. 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 less than 2 seconds, 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 4 times slower compared to SAH.

The genetic algorithm which is hybridized with DO2 (GAH) achieves better results (see Table 2 and Figure 1) compared to the simple genetic algorithm (GA) on all of the test cases. In all cases considered, the GAH out perform 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 2.55 seconds. GAH would be ideal for larger values of N where an optimal solution is not readily feasible. It is observed that averagely the GA takes lesser time to run compared to GAH.

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 33.1 times slower for PSO and 22.9 for PSOH in relation to SA). 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 1.45 times faster than PSO. While it is observed for all other meta-heuristics that as the number of machines increase their corresponding penalties reduce, the reverse is the case for PSO and PSOH.

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 (see Tables 1 &2 and Figures 1). On average, SA takes 0.45 seconds to run. However, it is about 4.41, 5.72, 33.1 and 22.9 times quicker than the GA, GAH, PSO and PSOH respectively. SA has the best overall time performance among all the meta-heuristics.

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 over all best results among the meta-heuristics in terms of performance in relation to penalty. The average timing is a little less than 0.5 seconds.
Due to equality of their variances, subsets of homogeneous groups are displayed in Table 2 using Scheffe's method. Three groups are obtained: group 1—SAH, GAH, SA and PSOH, group 2—PSO and group 3 – GA. These groups are arranged in decreasing order of their effectiveness. The worst among them is the GA.

6. Conclusion

We considered an identical machine problem with the objective of minimizing the number of early and tardy jobs. Six meta-heuristics which incorporates a fast greedy heuristic were suggested that gave promising results. Computational experiments and statistical analyses were performed comparing these algorithms. The SAH was the best among the various meta-heuristics. This research can be extended in several directions. First, these results could be compared with an optimal solution. Second, the environment with uniform or unrelated parallel machines can be a practical extension.

REFERENCES

[1] **Adamu, M.O. & Adewunmi, A.** 2012b. Minimizing the Weighted Number of Tardy Jobs on Multiple Machines: A Review. *Asian-Pacific Journal of Operational Research*. Submitted for publication.

[2] **Adamu, M. & Adewunmi, A.** 2013. Single Machine Review to Minimize Weighted Number of Tardy Jobs. *Journal of Industrial and Management Optimization*. Accepted for publication.

[3] **Garey, M.R. & Johnson, D.S.** 1979. Computers and Intractability, A Guide to the Theory of NP Completeness. Freeman, San Francisco.

[4] **Graham, R.L., Lawler, E.L., Lenstra, T.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.

[5] **Ho, J.C. & Chang Y.L.** 1995. Minimizing the Number of Tardy Jobs for *m* Parallel Machines. *European Journal of Operational Research*, 84, pp 343-355.

[6] **Süer, G.A., 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.

[7] **Süer, G.A.** 1997. Minimizing the Number of Tardy Jobs in Multi-Period Cell Loading Problems. *Computers and Industrial Engineering*, 33(3,4), pp 721-724.

[8] **Süer, G.A., Pico, F. & Santiago, A.** 1997. Identical Machine Scheduling to Minimize the Number of Tardy Jobs when Lost-Splitting is Allowed. *Computers Industrial Engineering*, 33 (1,2), pp 271-280.

[9] **Van Den Akker, J.M., Hoogeveen, J.A. & Van De Velde, S.L.** 1999. Parallel Machine Scheduling by Column Generation. *Operations Research*, 47(6), pp 862-872.

[10] **Chen, Z. & Powel, W.B.** 1999. Solving Parallel Machine Scheduling Problems by Column Generation. *INFORMS Journal on Computing*, 11(1), pp 78-94.

[11] **Liu, M. & Wu, C.** 2003. Scheduling Algorithm based on Evolutionary Computing in Identical Parallel Machine Production Line. *Robotics and Computer Integrated Manufacturing*, 19, pp 401-407.

[12] **M'Hallah, R. & Bulfin, R.L.** 2005. Minimizing the Weighted Number of Tardy Jobs on Parallel Processors. *European Journal of Operational Research*, 160, pp 471-484.

[13] **Sevaux, M. & Thomin, P.** 2001. Heuristics and Metaheuristics for a parallel Machine Scheduling Problem: A Computational Evaluation. *Proceedings of 4th Metaheuristics Inter. Conference*, pp 411-415.

[14] **Baptiste, P., Jouglet, A., Pape, C.L. & 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.

[15] **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).

[16] **Sevaux, M. & Sörensen, K.** 2005. VNS/TS for a Parallel Machine Scheduling Problem. MEC-VNS: *18th Mini Euro Conference* pm VNS.

[17] **Li., C.L.**, 1995. A Heuristic for Parallel Machine Scheduling with Agreeable Due Dates to Minimize the Number of Late Jobs. *Computers and Operations Research*, 22 (3), pp 277-283.

[18] **Hiraishi, K., Levner, E., & Vlach, M.** 2002. Scheduling of Parallel Identical Machines to Maximize the Weighted Number of Just-In-Time Jobs. *Computers and Operations Research*, 29, pp 841-848.

[19] **Sung, S.C. and Vlach, M.** 2001. Just-In-Time Scheduling on Parallel Machines. *The European Operational Research Conference*, Rotterdam, Netherlands.

[20] **Lann, A. & Mosheiov, G.** 2003. A Note on the Maximum Number of On-Time Jobs on Parallel Identical Machines. *Computers and Operations Research*, 30, pp 1745-1749.

[21] **Čepek, O. & Sung, S.C.** 2005. A Quadratic Time Algorithm to Maximize the Number of Just-In-Time Jobs on Identical parallel Machines. *Computers and Operational Research*, 32, pp 3265-3271.

[22] **Janiak, A., Janiak, W.A. & Januszkiewicz, R.** 2009. Algorithms for Parallel Processor Scheduling with Distinct Due Windows and Unit-Time Jobs. *Bulletin of the Polish Academy of Sciences Technical Sciences*, 57, (3), pp 209-215.

[23] **Adamu, M. & Abass O.** 2010. Parallel machine scheduling to maximize the weighted number of just-in- time jobs. *Journal of Applied Science and Technology*, vol. 15, no. 1 and 2, pp 27–34.

[24] **Adamu M, & Adewunmi A.** 2012a. Metaheuristics for Scheduling on Parallel Machines to minimize the Weighted Number of Early and Tardy Jobs. *International Journal of Physical Sciences*. 7(10): pp 1641- 1652.

TABLE 1: Homogeneous Subsets

PENALTY

Scheffe[a]

HEURISTICS	N	Subset for alpha = 0.05		
		1	2	3
SAH	20	28.4050		
GAH	20	30.5940		
SA	20	41.6130		
PSOH	20	47.1060		
PSO	20		69.4160	
GA	20			91.5840
Sig.		.147	1.000	1.000

Means for groups in homogeneous subsets are displayed.

a. Uses Harmonic Mean Sample Size = 20.000.

Table 2: Performance of Meta-Heuristics for different values of 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
	GA	113	119.38	126	105	110.84	116	96	103.14	109	92	98.68	106	88	93.26	101
		1937	2006.86	2122	1891	1961.52	2266	1828	1909.34	1985	1890	1928.72	2000	1922	1987.5	2094
	GAH	63	76.04	85	59	69.52	59	52	64.02	76	51	59.96	67	41	51.8	62
		2781	2904.34	3063	2641	2770.62	2938	2484	2628.76	2782	2485	61	2750	2531	2648.48	2813
	PSO	80	90.42	104	80	93.44	103	83	92.08	99	82	90.54	95	80	88	95
N=100		13734	14069.74	14453	13359	13795.38	14250	13406	13703.4	14141	13468	14622.4	20687	14187	15614.72	18344
	PSOH	65	74.96	85	68	82.88	90	64	85.5	95	74	83.82	93	72	80.48	87
		10093	10771.24	11765	9625	10251.9	11172	9375	9993.76	10922	9234	10029.34	10687	9531	10251.54	11109
	SA	74	81.68	91	69	75.84	81	59	67.46	76	54	62.06	69	47	54.36	64
		422	482.36	609	391	439.5	516	375	425.06	485	390	435.64	516	406	459.08	531
	SAH	67	78.58	93	56	66.7	75	47	58.32	71	41	53.54	61	35	46.28	57
		516	564.34	641	484	506.38	562	437	470.62	500	437	468.14	515	453	476.62	532

		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
	GA	105	112.74	119	94	100.46	111	78	89.28	96	75	82.8	92	72	77.14	85
		1922	2014.68	2640	1844	1948.48	2453	1844	1915.86	2297	1875	1922.2	1985	1890	1986.2	2625
	GAH	32	43	53	25	38.1	45	22	35.68	43	22	31.94	39	18	27.86	36
		2797	2889.16	3031	2625	2751.26	2938	2484	2591.58	2719	2454	2579.58	2734	2484	2606.54	2766
	PSO	57	69.68	82	54	71.4	87	65	73.6	83	62	72.36	81	62	72.02	79
N=200		14187	15639.1	17656	13953	15253.88	16884	13750	15079.68	16750	13875	15277.78	17157	14218	15587.76	17687
	PSOH	33	42.38	54	31	52.34	63	46	57.24	65	49	57.46	65	48	56.64	67
		9953	10685.86	11672	9500	10410.12	20562	9359	10099.74	18812	9343	10094.66	18422	9484	10341.26	19719
	SA	49	57.2	66	36	50.02	58	31	43.64	51	31	38.02	45	24	33.68	41
		421	470	532	406	436.52	485	375	424.3	500	390	434.74	500	406	457.3	547
	SAH	32	44.12	52	24	35.96	49	18	31.92	42	17	26.88	34	15	23.02	34
		531	553.74	610	468	500.9	547	437	464.44	515	437	466.26	437	453	479.92	547

		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
	GA	96	109.28	127	85	94.9	105	71	82.52	95	67	74.3	83	60	68.26	76
		1938	2031.28	3266	1844	1945.92	2422	1844	1918.78	2391	1828	1962.22	2766	1891	2003.44	2734
	GAH	13	21.06	29	12	18.72	30	8	15.94	24	8	14.94	23	3	11.48	21
		2765	2868.62	3047	2609	2745.68	2938	2500	2579.72	2656	2500	2561.46	2687	2485	2592.92	2688
	PSO	44	57.7	69	42	57.38	75	50	63.2	72	41	61.44	70	56	61.96	72
N=300		14093	15610.6	18469	13890	15334.38	17375	13765	15153.82	17062	13906	15309.38	17969	14218	15566.66	18500
	PSOH	13	21	32	15	30.74	47	18	35.98	45	12	38.34	51	10	37.78	48
		10125	10898.8	21672	9610	10250.84	14594	9266	9850.64	10750	9094	9849.82	10610	9078	10079.08	11203
	SA	32	42.26	56	28	34.84	44	19	28.36	36	18	24.88	35	13	20.1	29
		422	479.12	563	391	448.86	547	375	432.72	516	390	435.64	485	406	451.26	516
	SAH	14	21.6	32	13	18.04	29	7	13.86	21	4	11.96	19	2	8.4	16
		516	551.54	609	468	501.16	578	437	459.1	500	437	461.88	500	437	463.8	516

		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
	GA	94	106.38	118	81	93.22	107	63	79.1	93	62	71.3	83	57	64.7	74
		1953	2021.86	2312	1844	1974.82	3141	1875	1914.62	169	1844	1974.86	3016	1938	1993.8	2625
	GAH	3	8.9	15	2	8.32	15	0	6.6	18	0	4.84	12	0	3.16	13
		2766	2861	2938	2640	2716.32	2782	2500	2566.86	2672	2484	2560.86	2860	2515	2576.74	2641
	PSO	42	54.72	66	37	52.88	71	36	56.24	65	30	54.64	62	49	54.62	61
N=400		14157	15572.48	17468	13110	14119.14	16953	12906	13153.44	13516	12953	13333.5	13641	13265	13559.38	13906
	PSOH	3	9.46	16	10	18.26	29	11	23.76	37	12	27.26	39	1	25.84	40
		9921	10640.96	11672	9437	10122.72	10922	9297	9868.22	10579	9016	9887.2	11188	9437	10066.84	13297
	SA	23	43.2	42	18	25.88	33	14	20.28	33	8	15.92	24	6	12.58	22
		422	468.78	532	406	442.44	500	375	417.8	484	390	437.8	390	406	446.32	515
	SAH	3	9.32	16	1	8.22	16	0	5.74	15	0	3.64	10	0	2	9
		515	549.7	609	468	496.56	532	437	460.06	516	437	466.54	532	437	472.78	547

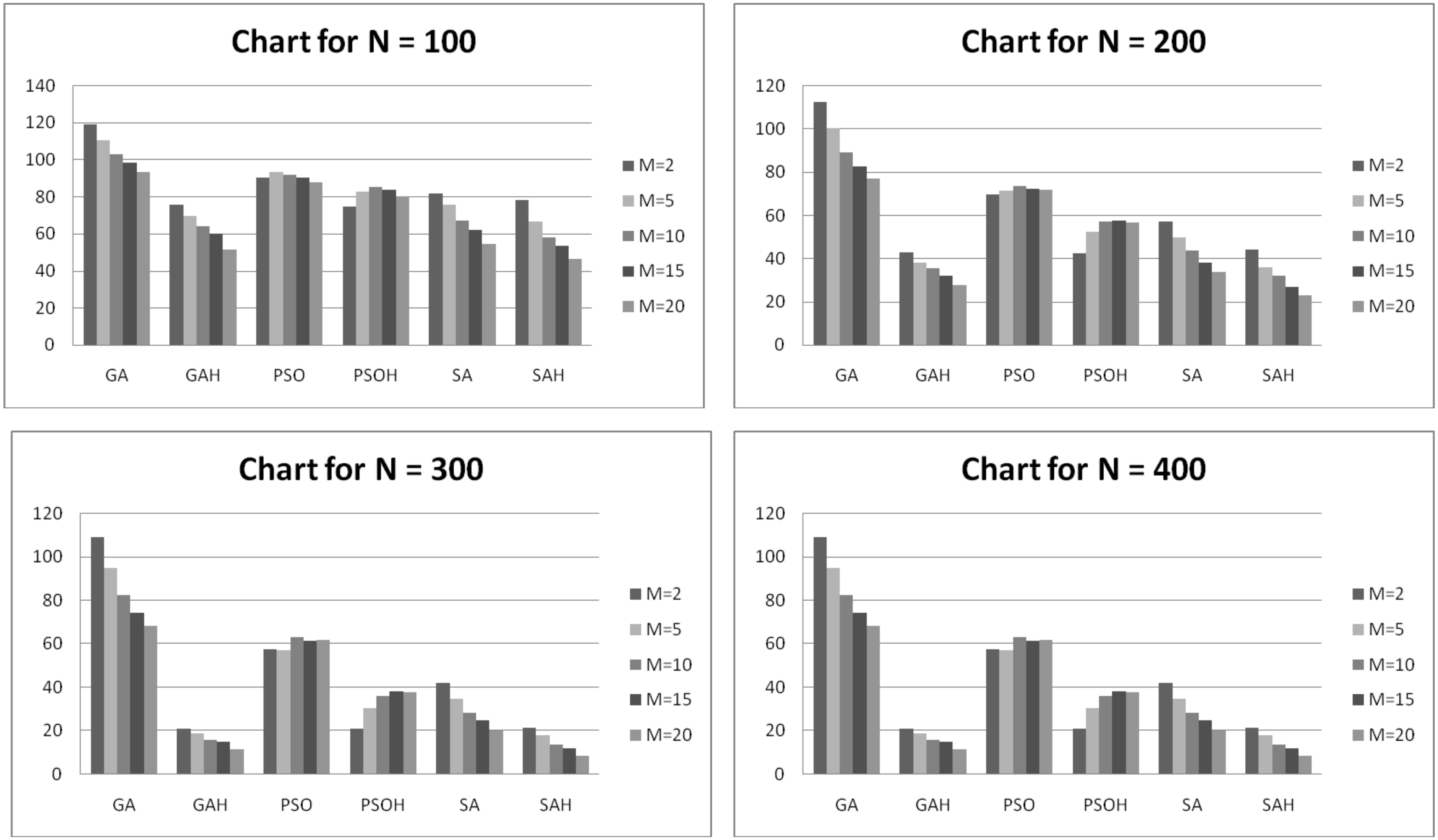

Figure 1: Meta-heuristics Performance in Relation to Penalty