Chapter 42
Identical Parallel-Machine Scheduling and Worker Assignment Problem Using Genetic Algorithms to Minimize Makespan

Imran Ali Chaudhry and Sultan Mahmood

Abstract Identical parallel machine scheduling problem for minimizing the makespan is a very important production scheduling problem which has been proven to be NP-hard. The problem further compounds with additional constraints. Genetic algorithms (GA) have shown great advantages in solving the combinatorial optimization problem in view of its characteristic that has high efficiency and that is fit for practical application. In this chapter we present a spreadsheet based GA approach for minimizing the makespan for scheduling of a set of tasks for identical parallel machines and worker assignment to machines. The results obtained from the proposed approach are compared with two sets of benchmark problems consisting of 100 problems each. It has been demonstrated that the performance of proposed approach is superior to the results that have been obtained earlier. The proposed approach produces optimal solution for almost 95% of the problems demonstrating the effectiveness of the proposed approach. An empirical analysis of GA parameters has also been carried out to see the effect on the performance of the proposed algorithm.

42.1 Introduction

Scheduling is an important process widely used in manufacturing, production, management, computer science, and among others. There are many kinds of scheduling problems in the literature. One of them is the parallel machine

I. A. Chaudhry (✉) · S. Mahmood
College of Aeronautical Engineering, National University of Sciences and Technology, Risalpur, Pakistan
e-mail: imran_chaudhry@yahoo.com

S. Mahmood
e-mail: sultanrandhawa@hotmail.com

scheduling (PMS) problem, in which scheduling jobs in parallel machines is considered. Many real life problems can be modeled as PMS ones. On production lines, it is common to find more than one machine of each kind carrying out the production tasks. Other examples are docks—ships, teachers student groups, hospital assistance—patients, etc.

Pinedo [1] pointed out that a bank of machines in parallel is a situation that is important from both a theoretical and a practical point of view. From a theoretical point of view, it is a generalization of the single machine and a special case of the flexible flowshop. From a practical point of view, it is important because the occurrence of resources in parallel is common in the real world. Also, techniques for studying machines in parallel are often used in decomposition procedures for multistage systems. PMS comes down to assigning each operation to one of the machines and sequencing the operations assigned to the same machine.

In particular, the makespan becomes an objective of considerable interest when dealing with machines in parallel. In practice, one often has to deal with the problem of balancing the load on machines in parallel; by minimizing the makespan, the scheduler ensures a good balance. Scheduling jobs in identical machines for minimizing the makespan was proved as NP-hard [2, 3]. Thus, it is unlikely to obtain the optimal schedule through polynomial-time-bounded algorithms.

In this chapter we present a spreadsheet based GA approach to minimize the makespan for scheduling a set of tasks on identical parallel machines and worker assignment to the machines.

42.2 Literature Review

The first work on PMS problems started at the end of the 1950s with McNaughton [4] and then with Hu [5]. These types of problems have received continuous interest from researchers since then, due to their relevance to computer systems and production systems and thus the literature continues to increase. Various constraints have been taken into account and different criteria have been studied. For a detailed review of new trends in PMS see Lam and Xing [6] while an extensive survey has been given by Cheng & Sin [7] and Moktoff [8].

Makespan minimization is one of the most frequently studied criteria in the scheduling literature and also in identical parallel machine scheduling. Some of the recent papers reviewed in the subsequent paragraphs are restricted to this criterion.

Min and Cheng [9] present a kind of GA based on machine code for minimizing the makespan in identical parallel machine scheduling. Hey demonstrate that the proposed GA is efficient ad fit for large scale problems and has advantage over heuristic procedure and simulated annealing method.

Yalaoui and Chu [10] consider real-life identical PMS problem with sequence-dependent setup times and job splitting to minimize makespan and propose a heuristic to solve this problem. Lee et al. [11] propose a simulated annealing method to generate near optimal solutions for the minimization of makespan in identical parallel machines. With the help of computational analysis they demonstrate that the proposed method is very accurate and outperforms the existing method. Akyol [12] also consider minimization of makespan in identical PMS and propose a Hopfield-like network that uses time-varying penalty parameters starting from zero and increases in a stepwise manner during iterations to overcome the tradeoff problem of the penalty function method.

Iori and Martello [13] consider scatter search algorithms and provide insights in the development of this heuristic technique and discuss the combinatorial difficulties of the problems through the analysis of extensive computational results. Akyol and Byhan [14] use neural networks for minimizing the maximum completion time (makespan) of jobs on identical parallel machines while Mokotoff et al. [15] propose algorithms that are based on list scheduling procedures.

Few other authors have considered other objectives along with minimization of makespan. Mohri et al. [16] consider minimizing two criteria, maximum completion time and maximum lateness for two and three identical PMS problems. Gupta and Ho [17] consider minimizing flowtime subject to optimal makespan on two identical parallel machines while Gupta and Ho [18] consider minimizing makespan subject to minimum flowtime on two identical parallel machines. Sricharoentham [19] consider memtic algorithm based on genetic algorithms for multiple objective scheduling in parallel machines.

Gupta et al. [20] consider makespan minimization on identical parallel machines subject to minimum total flow-time. Hong et al. [21] consider minimizing makespan of identical-machines scheduling problems with mold constraints. In this kind of problems, jobs are non-preemptive with mold constraints and several identical machines are available. They propose a GA based approach to solve this kind of problem. Ho et al. [22] present a two phase non-linear integer programming formulation for minimizing total weighted flowtime subject to minimum makespan on two identical parallel machines.

The papers reviewed above do not cater for the number of workers who are performing a certain task. Worker assignment scheduling problem has also been studied in the literature. In the classic PMS problem, no matter how many machines are involved, the number of workers at each machine may be ignored or assumed to be fixed and not taken into consideration. However, assigning more workers to work on the same job will decrease job completion time. Therefore, ignoring worker assignment decision may cause a managerial problem.

Hu [23, 24] considered the parallel machines models with decisions for job scheduling and worker assignment to minimize total tardiness and total flow time, respectively. A shortest processing time (SPT) heuristic and a largest marginal contribution (LMC) procedure were used to solve the job scheduling and worker assignment problems, respectively. The performance of these heuristics in Hu [23, 24] was further studied by Hu [25, 26], who concluded that these heuristics generate the same results no matter what the value of W (number of workers).

Chaudhry and Drake [27] also considered the minimization of total tardiness for the machine scheduling and worker assignment problems in identical parallel machines using GA. While Chaudhry [28] considered the minimization of total flow time for the worker assignment problem in identical parallel machine models using GA. Chaudhry et al. [29] proposed a GA to minimize makespan for machine scheduling and worker assignment problem in identical parallel machine models.

42.3 Problem Definition and Assumptions

In classical PMS problem, there are two essential issues to be dealt:

1. Partition jobs to machines.
2. Sequence jobs for each machine.

However, in the present research, the worker assignment scheduling problem needs to solve two sub-problems: how to assign jobs to machines and workers to machines? The objective is to minimize the makespan, which is defined as the total completion time of all the jobs. The performance of GA is compared with SPT/L(east)PA and L(argest)PA heuristics heuristic approach by Hu [30].

We assume that all machines are identical such that the processing time of a job is independent of machine. Deterministic processing times and due dates are assumed. Machine setup times are included in the processing time. The only difference between worker assignment scheduling problem and the classic scheduling problem is that the former has an additional constraint of worker assignment also. The processing times of the jobs are therefore dependent on the number of workers assigned to work on a particular machine.

The following notation is used to define the problem.

A_i, B_i, E_i	Integers, follow uniform distribution, such that $0 \leq A_i < 10$, $0 \leq B_i \leq 50$, $0 \leq E_i \leq 10$ (1st data set) $0 \leq A_i < 10$, $0 \leq B_i \leq 800$, $0 \leq E_i \leq 10$ (2nd data set)
n	Number of jobs
m	The number of parallel machine in the shop
p_i	The process time of job i
W	Number of workers in the shop
Wj	Number of workers assigned to machine m_j

We have to assign n jobs to m parallel machines taking into account the following characteristics:

1. Each job has only one operation.
2. The jobs have different processing times which depend on the number of workers assigned to that machine.
3. No job pre-emption/splitting is allowed.
4. Each job has its own due date.

5. The selected objective is to minimize the total tardiness.
6. Any machine can process any job.
7. Machine setup times are negligible.
8. No machine may process more than one job at a time.
9. Transportation time between the machines is negligible.
10. Number of jobs and machines are fixed.
11. There is a group of W workers that have the same abilities to perform the duties assigned to them.
12. All m machines, n jobs and W workers are available at time zero.
13. Assume processing time function has a simplified form of $p_i(W_j) = A_i + B_i/(E_i \times W_j)$ where $p_i(W_j)$ is the processing time of job I that is processed on machine j in which the number of W_j workers are assigned on it, A_i is a fixed constant and not affected by the number of workers, $B_i/(E_i \times W_j)$ is the variable part and affected by the number of workers.
14. The number of workers assigned to each machine needs to be decided before any job can be processed and they will not be re-assigned until all the jobs have been completed.

The simulation experiments have been carried out for 12 jobs to be scheduled on three parallel machines with 10 workers.

42.4 Genetic Algorithms

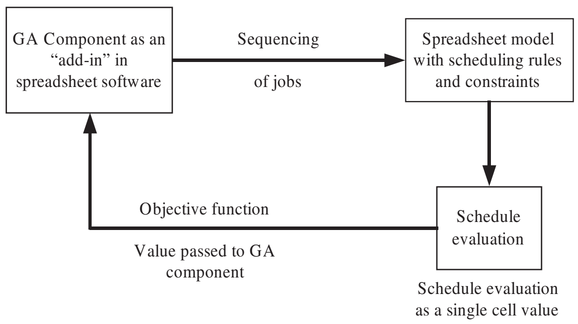
GAs is one of problem solving systems based on the principles of evolution and hereditary, each system start with an initial set of random solutions and use a process similar to biological evolution to improve upon them that encourage the survival of the fittest. The best overall solution becomes the candidate solution to the problem. A detailed introduction to Gas is given in Goldberg [31]. The earliest application of GA has been reported by Davis [31]. For a recent review of GA application in scheduling is given in Chaudhry and Drake [27] and Chaudhry [28].

In this study, the tool use for carrying out the GAs is a commercial software package called EVOLVER^{TM} [32], which functions as an add-into Microsoft Excel^{TM}. Evolver has been used for the purpose of production scheduling recently by number of researchers [27–29, 33–35].

The objective function, variables (adjustable cells), and the constraints are readily specified by highlighting the corresponding spreadsheet cells. The scheduler/evaluator portion of the model is constructed by using the spreadsheet's built in function. The schematic in Fig. 42.1 illustrates the integration of the GA with the spreadsheet.

The advantage of the proposed method is that the program runs in the background freeing the user to work in the foreground. When the program finds the best result it notifies the user and places the values into the spreadsheet for analysis. This is an excellent design strategy given the importance of interfacing with spreadsheet in business.

Fig. 42.1 Integration of GA and spreadsheet


GA Component as an "add-in" in spreadsheet software
Sequencing of jobs
Spreadsheet model with scheduling rules and constraints
Schedule evaluation
Objective function Value passed to GA component
Schedule evaluation as a single cell value

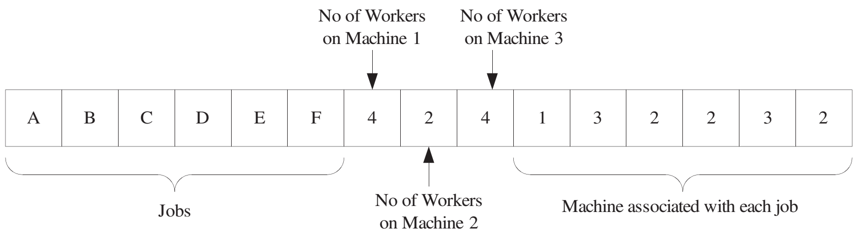

No of Workers on Machine 1
No of Workers on Machine 3
A
B
C
D
E
F
4
2
4
1
3
2
2
3
2
Jobs
No of Workers on Machine 2
Machine associated with each job

Fig. 42.2 Chromosome representation

42.4.1 Chromosome Representation

The chromosome representation for the machine scheduling and worker assignment problem is shown in the Fig. 42.2.

For determining the ordering of the jobs i.e., for the first six genes we needed to handle the permutation representation. The next three genes represent the number of workers assigned to each machine stating that four workers are assigned to machine 1, two workers to machine 2 and four workers to machine 3. While the machine assignment block can be read as the assignment of jobs to each of the six machines, whereby the assignment of the jobs to machines would be as follows: Job A to be processed on machine 1; job B on machine 3, job C on machine 2, job D on machine 2, job E on machine 3 and job F on machine 2, respectively.

Each of the block in Fig. 42.2 is calculated at a different location in the spreadsheet which is in turn linked to the calculation of the objective function i.e., the makespan.

42.4.2 Reproduction/Selection

Evolver uses a steady state approach. This means that only one organism rather than an entire population is replaced at a time. The number of generations can be found by dividing the trials by the size of the population. As far as parent selection

Fig. 42.3 Steady state algorithm

```
Repeat
    Create n children through reproduction
    Evaluate and insert the children into the population
    Delete the n members of the population that are least fit
Until stopping criteria reached
```

is concerned, in Evolver, parents are chosen using a rank-based mechanism. This procedure begins by rank ordering the population by fitness. Next an assignment function gives each individual a probability of inclusion. The assignment function can be linear or nonlinear. A Roulette wheel is then built with the slots determined by the assignment function. The next generation of an n-sized population is built by giving the wheel n spins. This procedure guides selection towards the better performing members of the population but does not force any particular individual into the next generation. Figure 42.3 describes the steady state algorithm.

42.4.3 Crossover Operator

As for the first six genes of the chromosome, we needed to handle permutation representation; the "order solving method" of Evolver was used. This method applies order crossover operator [36]. This selects items randomly from one parent, finds their place in the other parent, and copies the remaining items into the second parent in the same order as they appear in the first parent. This preserves some of the sub-orderings in the original parents while creating some new sub-orderings. Figure 42.4 shows the crossover operator as described above.

For the number of workers and machine assignment the "recipe solving method" of Evolver was used. The "recipe solving method" implements uniform crossover [36]. This means that instead of chopping the list of variables in a given scenario at some point and dealing with each of the two blocks (called "single-point" or "double-point" crossover), two groups are formed by randomly selecting items to be in one group or another. Traditional x-point crossovers may bias the search with the irrelevant position of the variables, whereas the uniform crossover method is considered better at preserving schema, and can generate any schema from the two parents. For worker assignment (genes 7–9 in Fig. 42.2) and machine assignment (genes 10–15 in Fig. 42.2), we have a set of variables that are to be adjusted and can be varied independently of one other. In the spreadsheet model presented in this chapter, the constraint placed on the workers and machines is to set the range that the variable must fall between. Similarly, as there are ten workers available in the shop, another constraint that ensures that only valid solutions are retained in the population is that the sum of values generated for genes 7–9 should be equal to 10. This ensures that no invalid solution is retained in the population pool. Figure 42.5 shows a typical uniform crossover operator.

Fig. 42.4 Order crossover operator

Position:	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***
Parent 1 (P1):	1	2	3	4	5	6	7	8	9
Binary Template:	*0*	*1*	*1*	*0*	*1*	*1*	*0*	*0*	*1*
Parent 2 (P2):	4	9	5	8	3	6	7	1	2
Offspring (O):	4	2	3	8	5	6	7	1	9

Fig. 42.5 Uniform crossover—a given % of the organism is randomly selected

42.4.4 *Mutation Operator*

To preserve all the original values, the "order solving method" performs mutation by swapping the positions of some variables in the organism. The number of swaps performed is increased or decreased proportionately to the increase and decrease of the mutation rate setting (from 0 to 1).

The "recipe solving method" performs mutation by looking at each variable individually. A random number between 0 and 1 is generated for each of the variables in the organism, and if a variable gets a number that is less than or equal to the mutation rate (for example, 0.06), then that variable is mutated. The amount and nature of the mutation is automatically determined by a proprietary algorithm. Mutating a variable involves replacing it with a randomly generated value (within its valid min–max range).

42.5 Experimental Results

42.5.1 *Implementation Details*

The job order (first six genes) in Fig. 42.2 is a permutation of a list of jobs where we are trying to find the best way to arrange a set of given jobs. This permutation is independent of the number of workers on each machine and assignment of job to a particular machine. However, the objective function is calculated keeping in view all the constraints which are discussed in the next paragraph.

For number of workers on each machine and the machine corresponding to each job, i.e., for genes 7–9 and 10–15 (Fig. 42.2), random integer numbers are generated by the GA subject to the constraints that have been defined in the initial setup of the GA. Constraints are basically the conditions that must be met for a solution to be valid. The constraint imposed on the number of workers on each machine is that the sum of all workers assigned to the three machines should always be 10, which is the total number of workers available in the shop. The range for random integer is from 1 to 10. Therefore, only those solutions are kept in the population where the sum of all workers is 10. Hence, only integer values between 1 and 10 (both limits inclusive) are generated, while the constraint for machine corresponding to each job is that an integer number is selected from among 1, 2, and 3 (which are machine numbers).

42.5.2 Computational Analysis

In order to check the effectiveness of the proposed spreadsheet based GA approach two data set of 100 problems each [30] were used. The problems in both the data sets have 12 jobs to be scheduled on three identical parallel machines where the number of workers available is equal to 10. The only difference among the two data sets in terms of the value of the variable B_i, for first data set it is $0 \leq B_i \leq 50$ and for second data set it is $0 \leq B_i \leq 800$. The problems have been simulated on a P1V 1.7 GHz computer having 512 MB RAM. By conducting repeated tests, we found that the best values to be set for the number of population, the crossover rate, and the mutation rate are 60, 0.65, and 0.09, respectively. Therefore, for each of the run, same set of parameter setting as described previously have been used, which correspond to 3 min 20 s on a PIV 1.7 GHz computer having 512 MB RAM.

Tables 42.1 and 42.2 give a summary of the results for the first and second data sets, respectively. The summary shows the performance of proposed GA approach with the SPT/L(east)PA and L(argest)PA heuristic proposed by Hu [30].

GA produced superior solution as compared to the SPT/L(east)PA and L(argest)PA heuristic. For the first data set GA found 95 optimal solutions as compared to the heuristic which found optimal solution for 22 problems. While for second data set these values were 96 and 27, respectively. The average time for the GA for first and second data set was to find the best solution was 70 and 86 s, respectively. As compared to the optimal solution, for the first data set the maximum percentage error was 7.87 and 65.47% for the GA and SPT/L(east)PA and L(argest)PA heuristic [30], respectively. While for the second data set these maximum percentage errors were 8.94 and 38.31%, respectively.

As compared to SPT/L(east)PA and L(argest)PA heuristic, for the first data set GA produced same solution for 22 problems and better for 77 problems. For the

Table 42.1 Summary of results for the two approaches—1st data set

		GA	SPT/L(east)PA and L(argest)PA heuristic [30]
# of problems simulated		100	100
# of optimal solution obtained		95	22
Max percentage error (%)		7.87	65.47
Avg Time to find best solution		70 s	–
Comparison of GA with other methods	Same	–	22
	Better	–	77
	Worse	–	1

Table 42.2 Summary of results for the two approaches—2nd data set

		GA	SPT/L(east)PA and L(argest)PA heuristic [30]
# of problems simulated		100	100
# of optimal solution obtained		96	18
Max percentage error (%)		8.94	38.31
Avg Time to find best solution		86 s	–
Comparison of GA with other methods	Same	–	27
	Better	–	71
	Worse	–	2

Table 42.3 Combinations of GA parameter levels

Population size	20, 40, 60, 80
Mutation rate	0.01, 0.03, 0.05, 0.07, 0.09
Crossover rate	0.2, 0.4, 0.6, 0.8

second data set these were 27 and 71, respectively. While for only one problem in first data set and two problems in second data set GA produced worse results as compared to SPT/L(east)PA and L(argest)PA heuristic.

42.6 Empirical Analysis of the Effect of GA Parameters

The performance of a GA depends upon population size, crossover, and mutation rate. Detailed experiments have been carried out to study the effect of these three parameters. The application of the GA to the schedule optimization is repeated here for each of the combinations of the parameter levels given in Table 42.3 for both the data sets mentioned in Sect. 42.5.2.

For each combination of the parameters, the GA is run for ten different random initial populations. These ten populations are different for each combination. Thus, in total, the GA is run 800 times. The values chosen for the population size are representative of the range of values typically seen in the literature, with 0.09 being included to highlight the effect of a relatively large mutation rate. The crossover rates chosen are representative of the entire range.

The results show that the performance of the GA is insensitive to the crossover rate. There is also a degree of insensitivity to the mutation rate, which possessed a ‘range’ over which its value is suitable rather than a more precise value. Furthermore, performance is fairly insensitive to population size, provided the mutation rate is in the ‘good’ range. The various set of problems that have been solved in this chapter demonstrate that GA-spreadsheet approach presented in this chapter performs well on a wide range of shop models with a ‘general purpose’ set of parameters.

42.7 Conclusions

This chapter presented a spreadsheet based general purpose GA solution methodology for the scheduling of set of job on parallel machines and worker assignment to machines. The spreadsheet GA implementation has been found to be easy to implement catering for the peculiarities of any environment. Moreover, the spreadsheet environment makes it very suitable to carry out what if analysis. The results in Tables 42.1 and 42.2 clearly show the superiority of proposed GA approach as compared to an earlier study by Hu [29]. The spreadsheet model can be easily customized to include additional jobs, machines or workers without actually changing the logic of the GA routine thus making it a general purpose scheduling approach.

The key advantage of GAs portrayed here is that they provide a general purpose solution to the scheduling problem which is not problem-specific, with the peculiarities of any particular scenario being accounted for in fitness function without disturbing the logic of the standard optimization routine. The GA can be combined with a rule set to eliminate undesirable schedules by capturing the expertise of the human scheduler.

The empirical analysis of the GA parameters shows that crossover rate is often an insignificant factor in the performance of the GA. There is a degree of insensitivity to the mutation rate in so far as the ‘good’ values form a fairly broad range rather than coming at a more precise point. However, outside of the ‘good’ range performance soon deteriorates greatly. It is observed that a low mutation rate is desirable when the population size is large and a higher mutation rate is desirable when the population size is small. The sensitivity to population size is greatly reduced when the mutation rate is in the ‘good’ range. As a consequence of these findings, it has been argued that GAs have the potential to make a general-purpose real-world scheduler.

References

1. Pinedo ML (2008) Scheduling theory algorithms and systems. Springer Science, New York
2. Brucker P (1995) Scheduling algorithms. Springer, New York
3. Garey MR, Johnson DS (1979) Computers and intractability a guide to the theory of NP-completeness. Freeman, San Francisco
4. McNaugton R (1959) Scheduling with deadlines and loss functions. Manag Sci 6:1–12
5. Hu TC (1961) Parallel sequencing and assembly line problems. Oper Res 9:841–948
6. Lam K, Xing W (1997) New trends in parallel machine scheduling. Int J Oper Prod Man 17(3):326–338
7. Cheng T, Sin C (1990) A state-of-the-art review of parallel machine scheduling research. Eur J Oper Res 47:271–292
8. Mokotoff E (2001) Parallel machine scheduling problems: a survey. Asia Pac J Oper Res 18:193–242
9. Min L, Cheng W (1999) A genetic algorithm for minimizing the makespan in the case of scheduling identical parallel machines. Artif Intell Eng 13(4):399–403

10. Yalaoui F, Chu C (2003) An efficient heuristic approach for parallel machine scheduling with job splitting and sequence-dependent setup times. IIE Trans 35(2):183–190
11. Lee W-C, Wu C-C, Chen P (2006) A simulated annealing approach to makespan minimization on identical parallel machines. Int J Adv Manuf Tech 31:328–334
12. Akyol DE (2007) Identical parallel machine scheduling with dynamical networks using time-varying penalty parameters. In: Levner E (ed) Multiprocessor scheduling: theory applications. Itech Education and Publishing, Vienna, pp 293–314
13. Iori M, Martello S (2008) Scatter search algorithms for identical parallel machine scheduling problems. Stud Comp Intell 128:41–59
14. Akyol DE, Bayhan GM (2006) Minimizing makespan on identical parallel machines using neural networks. Lect Notes Comput Sci 4234:553–562
15. Mokotoff E, Jimeno JL, Gutiérrez AI (2001) List scheduling algorithms to minimize the makespan on identical parallel machines. TOP 9(2):243–269
16. Mohri S, Masuda T, Ishii H (1999) Bi-criteria scheduling problem on three identical parallel machines. Int J Prod Econ 60–61:529–536
17. Gupta JND, Ho JC (2000) Minimizing flowtime subject to optimal makespan on two identical parallel machines. Pesqui Oper 20(1):5–17
18. Gupta JND, Ho JC (2001) Minimizing makespan subject to minimum flowtime on two identical parallel machines. Comput Oper Res 28(7):705–717
19. Sricharoentham C (2003) Multiple-objective scheduling in parallel machines using memtic algorithm. MS Thesis Texas Tech University, USA
20. Gupta JND, Ho JC, Ruiz-Torres AJ (2004) Makespan minimization on identical parallel machines subject to minimum total flow-time. J Chin Inst Ind Eng 21(3):220–229
21. Hong TP, Sun PC, Jou SS (2009) Evolutionary computation for minimizing makespan on identical machines with mold constraints. WSEAS Trans Syst Control 7(4):339–348
22. Ho JC, López FJ, Ruiz-Torres AJ, Tseng TL (2009) Minimizing total weighted flow-time subject to minimum makespan on two identical parallel machines. J Intell Manuf doi:10.1007/s10845-009-0270-1
23. Hu PC (2004) Minimizing total tardiness for the worker assignment scheduling problem in identical parallel-machine models. Int J Adv Manuf Tech 23(5–6):383–388
24. Hu PC (2005) Minimizing total flow time for the worker assignment scheduling problem in the identical parallel-machine models. Int J Adv Manuf Tech 25(9–10):1046–1052
25. Hu PC (2006) Further study of minimizing total tardiness for the worker assignment scheduling problem in the identical parallel machine models. Int J Adv Manuf Tech 29:165–169
26. Hu PC (2006) Further study of minimizing total flowtime for the worker assignment scheduling problem in the identical parallel machine models. Int J Adv Manuf Tech 29(7–8):753–757
27. Chaudhry IA, Drake PR (2008) Minimizing total tardiness for the machine scheduling and worker assignment problems in identical parallel machines using genetic algorithms. Int J Adv Manuf Tech 42:581–594
28. Chaudhry IA (2010) Minimizing flow time for the worker assignment problem in identical parallel machine models using GA. Int J Adv Manuf Tech 48(5–8):747–760
29. Chaudhry IA, Mahmood S, Ahmad R (2010) Minimizing makespan for machine scheduling and worker assignment problem in identical parallel machine models using GA. Lecture notes in engineering and computer science: Proceedings of the world congress on engineering 2010, WCE 2010, 30 June–2 July, London, UK, pp 2464–2469
30. Hu PC Online research data set available at http://sparc.nfu.edu.tw/ ~ pchu/research_data.htm
31. Goldberg DE (1989) Genetic algorithms in search optimization and machine learning. Addison-Wesley, Boston
32. Palisade Corporation (1998) Evolver: the genetic algorithm super solver. New York, USA
33. Chaudhry IA, Drake PR (2008) Minimizing flow time variance in a single machine system using genetic algorithms. Int J Adv Manuf Tech 39:355–366

34. Hayat N, Wirth A (1997) Genetic algorithms and machine scheduling with class setups. Int J Comput Eng Manag 5(2):10–23
35. Ruiz R, Maroto C (2002) Flexible manufacturing in the ceramic tile industry. In: Proceedings of eighth international workshop on project management and scheduling, April 3–5, Valencia Spain
36. Davis L (1991) Handbook of genetic algorithms. Van Nostrand Reinhold, New York