Volume 2 No.6, June 2012 ISSN 2224-3577
International Journal of Science and Technology

http://www.ejournalofsciences.org

Minimizing Makespan in Parallel Scheduling Problem with Deterioration using two Meta Heuristics

Hamidreza Haddad[1], Dariush Moradinezhad[2]
[1]Department of Industrial Engineering, IranUniversity of Science and Technology, Narmak, Tehran, Iran
[2]Department of Industrial and Manufacturing Engineering, Western Michigan University Kalamazoo, MI, USA

ABSTRACT

This paper tackles a parallel scheduling problem, with considering deterioration to minimize make-span function. A mathematical model is developed and because of high complexity, two meta-heuristics including the extended compact genetic algorithm (ECGA) and simulated annealing (SA) are used to obtain near optimal solutions in reasonable run time.
For the computational measures, the sensitivity analysis is implemented for deterioration, and various instances are presented that show the effectiveness and capability of proposed methods.

KEYWORDS: *parallel machine scheduling; deterioration; makespan*

1. INTRODUCTION

Parallel machine scheduling is one of the oldest problems in the scheduling area that is applied widely in manufacturing and real industry.

Parallel machine scheduling has practical application in wafer fabrication process, testing process of electrical circuits, chemical processes performed in tanks or kilns, and heat-treating ovens. The assignment of an item to those machines is somehow different form classical assignment. At first, the batches are formed and the items are allotted to them, and then, they need to be assigned to the machine in order to optimize the scheduling criteria such as make-span, total tardiness, etc. The batch-processing machine problems might be classified into three major categories according to the batch processing time. The first is called *fixed batch* model in which the items that a batch contains have identical size, so the batch processing time is equal to processing time of each of the items and is constant. In this case, the number of items assigning to a batch is limited [2, 3]. The second is called *burn-in* model, in which the processing time of a batch depends on the size of the jobs constitute it [4, 5]. In this case, the longest processing time of the jobs which the batch includes determines the batch processing time. In these two discussed case, the machine is capable of processing several items simultaneously, and also all jobs assigned to a batch have identical starting and completion times. The third one is *serial batching* model, in which the machine processes only one item at the time, and the processing time of the batch is equal to sum of the processing times of the jobs that the batch contains [6]. Ikura and Gimple(1986)[7] proposed a fixed batch model in which dynamic job arrivals is considered, and an optimal algorithm to minimize the makespan is suggested. Considering a single batch processing machine model, Chandru et al. (1993a) [8] developed a branch-and-bound procedure to obtain exact solution, but the algorithm handles the small instances. Dobson and Nambimadom (1992, 2001) [9,10] and Uzsoy (1994, 1995) [11,12], considering non-identical job sizes, proposed their single batch processing models to minimize the total weighted completion time ($\sum w_i C_i$), minimizing total completion time ($\sum C_i$), and maximum lateness ($L_{\max}$) and also total weighted completion time, respectively. Also, Kempf et al. (1998) [13] suggested a single batch processing models to minimize the total completion time and makespan under constraint of the board availability as well as oven capacity. Dupont and Dhaenens-Flipo[14]proposed a branch-and-bound algorithm in order to minimize makespan of a single batch-processing machine. Melouk et al. [15]applied a SA approach to minimize the makespan of a single batch-processing machine considering non-identical job sizes. Damodaran et al. [16] used a simulated annealing approach and a genetic algorithm to minimize the makespan of the single batch processing machine model. Kashan et al. [17] solved the single batch processing machine model with non-identical job size applying two different genetic algorithms in order to minimize the makespan. Considering parallel machines, Chandru et al. 1993a, Uzsoy 1995, Hochbaum and Landy 1997, and Chandra and Gupta 1997 have proposed their batch processing models considering identical job size [18, 19, 20, 21]. Koh et al [22] examined parallel batch processing machines with non-identical job size and incompatible job batches to minimize makespan, total completion time, and total weighted completion time using some heuristics and genetic algorithm. Chang, Damodaran, and Melouk [23] developed a SA algorithm to obtain the minimized makespan on parallel batch processing machines. Lin and Jeng [24] utilized a dynamic programming algorithm to determine the minimum of L_{max} and the number of tardy jobs in a parallel batch processing machine problem. Malve and Uzsoy [25] considered dynamic job arrivals in a parallel batch processing machine to minimize L_{max}. They also, suggested some iterative improvement heuristics and integrated them using a genetic algorithm. Assuming non-zero job preparation times, Chung et al. [26] suggested a model and three heuristics to minimize the makespan. Damodaran et al. [27] developed a Greedy Randomized adopted Search Procedure (GRASP) approach and

showed how it outperforms other non-exact procedures to minimize the makespan. Some studies are directed on flow shops that contain one or more batch processing machines. Ahmadi et al. (1992) [28] analyzed a class of problems in which there are a two or three machine flow shops, where one of the machines is a batch processing machine. Sung and Kim [29] suggest a two-batch-machine flow shop scheduling problem to minimize maximum tardiness, total tardiness and number of tardy jobs. Mosheiov et al. [30] introduced a solution procedure for batch scheduling on an *m*-machine flow shop minimizing flow time where jobs have equal processing time and setup times are assumed to be independent of the sequence and machine. Lin et al. [31] presented a three-machine assembly-type flow shop model considering identical processing time on the batch processor stage while aiming at the minimization of makespan.

The remainder of this paper is organized as follows: The notation and problem description are introduced in section 2. Section 3 presents the solution approaches. In section 4, computational experiments are reported and statistical analysis is performed. Finally, some concluding remarks are made in section 5.

2. PROBLEM FORMULATION

In this paper, the problem of sequencing of N jobs on M parallel machines with deteriorating jobs is considered. Each machine works with no idle and all the jobs are ready to be processed at time zero.

The aim is to find a schedule of jobs that minimizes makespan. For this regard a mathematical model is presented.

According to deterioration, the processing times of jobs have not constant values and are dependent to their positions. We use of the time dependent deterioration model investigated by Wang et al [37] that is presented as:

$$p_{[i]} = p_i(1 + p_1 + p_2 + \cdots + p_{[i-1]})^{\delta} \qquad (1)$$

Where $0< \delta <1$ and $p_{[1]} = p_1$.

The variables and parameters of problem are defined as follows:

N	Number of jobs are ready to scheduled
M	Number of machines
p_i	The normal processing time of job where scheduled in i-th position
$p_{[i]}$	The actual processing time of job where scheduled in i-th position
$C_{j,k}$	The completion time of j-th position on machine k
δ	Rate of deterioration

And decision variables are as follows:

$$x_{ijk} = \begin{cases} 1 & \text{If job i is executed in priority j on machine k} \\ 0 & \text{Otherwise} \end{cases}$$

And the proposed model is as follows:

$$Min \quad \max\{c_{j,k}\} \qquad \text{J=1,2,..,N; K=1,2,…,M} \qquad (2)$$

St:

$$\sum_j \sum_k x_{ijk} \le 1 \qquad \text{J=1,2,..,N; K=1,2,…,M} \qquad (3)$$

$$\sum_i x_{ijk} = 1 \qquad \text{i=1,2,…,N} \qquad (4)$$

$$c_{0,k} = 0 \qquad \text{K=1,2,…,M} \qquad (5)$$

$$c_{j,k} = c_{j-1,k} + \sum_{i=1}^{n} x_{ijk} P_{[i]} \qquad \text{J=1,2,..,N} \qquad (6)$$

Equation (2) introduces the objective function and tries to find a schedule of jobs that minimizes makespan. Constraint (3) states that in each priority of machines just one job could be planned. Constraint (4) assures that each job must be processed only one time. Constraint (5) mentions that machine is available from time zero. Constraint (6) declares that how the value of completion time for each machine is calculated.

3. SOLUTION APPROACH

3.1 Extended Compact Genetic Algorithm

ECGA, proposed by Harik (1999) is based on a key idea that the choice of a good probability distribution is equivalent to linkage learning. Linkage learning can be considered as identifying blocks. The measure of a good distribution is quantified based on minimum description length (MDL) models. The key concept behind MDL models are that

given all things are equal, simpler distributions are better than the complex ones.

CGA views the GA population as a vector of probability distributions and the crossover as a sampling operation on the distributions. ECGA extend the probability model in CGA from a probability vector to the marginal product model (MPM). MPMs are similar to the models employed by CGA and PBIL; expect that they can represent the joint probability distribution over more than one gen at a time.

The object of ECGA is to find "good" distributions. Good distributions are those under which the representation of the distribution using the current encoding, along with the representation of the population compressed under that distribution, is minimal. One way to realize this concept is the minimum description length (MDL).

A flowchart of ECGA procedure is given in fig 1. Two things need further explanation, one is the identification of MPM using MDL and the other is the creation of a new population based on MPM. The identification of MPM in every generation is formulated as a constrained optimization problem, so that combined complexity would be minimum.

$$combined\ complexity \quad C_m + C_p \tag{7}$$

$$C_m = Model\ Complexity = \log N \sum_{i=1}^{m} 2^{si} \tag{8}$$

$$C_p = Compressed\ Population\ Complexity = N \sum_{i=1}^{m} \sum_{p} - p \log_2 p \tag{9}$$

Where m is the number of groups, s_i is the size of i th group, p is the probability of an allele pattern in i-th group, and N is the population size. The combined complexity is the summation of the model complexity and the compressed population complexity.

As an example, a simple MPM is shown in Table 1. MPM divides the genes or variables into several groups. In Table 1, four genes are divided into three groups [gene 0, gene 3], [gene 2], and [gene 1]. For each group, we count the occurrence of different patterns in the whole population and store it in the table. For gene 2, the number of 1's and the number of 0's are the same. We choose the MPM for two reasons: 1) they make the exposition simpler; and 2) the structure of the model can be directly translated into a linkage map.

Table 1: An Example MPM for Four Genes

Group 1 [0 3]		Group 2 [1]		Group 3 [2]	
allele	prob.	allele	prob.	allele	prob.
00	0.1	0	0.5	0	0.6
01	0.3	1	0.5	1	0.4
10	0.2				
11	0.4				

3.2 Simulation Annealing

Simulated annealing (SA) is a class of optimization Meta heuristics that performs a stochastic neighborhood search through the solution space that have been applied widely to solve many combinatorial optimization problems. The immense advantage of SA over classical a local search method is its ability to avoid getting trapped in local optima while searching for a global optimum.

In this case, SA starts with a randomly generated solution and neighborhood search is also implemented by swapping the two randomly selected positions in the current solution string.

4. COMPUTATIONAL RESULTS

To illustrate the efficiency and performance of the proposed methods, the algorithm procedure was coded in Visual Basic 6 and was run on a Vostro 1500 with 2.2 GHz CPU and 2 GB Ram. All the instances were randomly generated as follows. For each job, an integer processing time was generated from a uniform distribution [1, 100]. The results of the ECGA algorithm are compared with SA, then the sensitivity analyze of problem is checked to the deterioration rate. Table 3 illustrates the comparison of the ECGA algorithm with SA for different deteriorating rates. It's clear that the ECGA algorithm acts much better than the simple SA.

Table 2: Comparison between the Results of ECGA and SA

N	M	δ	ECGA		SA	
			VOF	Time (Sec)	VOF	Time (Sec)
5	2	0.2	34142	0	34142	0
	3	0.4	43622		43622	
	5	0.8	18742		20831	
10	2	0.2	341201	0	363101	0
	4	0.4	359100		389177	
	6	0.8	272732		304645	
25	2	0.2	3172107	5	3201679	2
	4	0.4	3305152		3353239	
	10	0.8	3551189		3687950	
50	2	0.2	17512709	45	17512815	20
	6	0.4	15217431		16191189	
	18	0.8	13447632		14428721	
75	5	0.2	28455311	245	37799277	122
	10	0.4	40012023		43795066	
	20	0.8	47812415		59054645	

Where N demonstrates the number of jobs, M shows the number of machines and δ represents the rate of deterioration. Columns 3 and 4 also show the performance of ECGA include value of objective function (VOF) and running time and columns 5 and 6 represent the results of SA.

Furthermore, the performance of ECGA and SA are tested by consider of several rate of deteriorations with choose of medium scale of jobs and the results are depicted in table 4.

Table 3: Sensitivity Analyze based on Deterioration Rate

δ	ECGA	SA
	VOF	VOF
0.1	2904959	3509469
0.2	3049109	3690822
0.3	2469869	3372463
0.4	3759849	4312127
0.5	3467239	4711154
0.6	2980549	4392259
0.7	4039459	5759277
0.8	3944670	6623464
0.9	3345769	5187022

Where the value of deterioration rate is shown by δ and second and third columns show the performance of presented methods. In order to better view, this comparison is depicted in below figure.

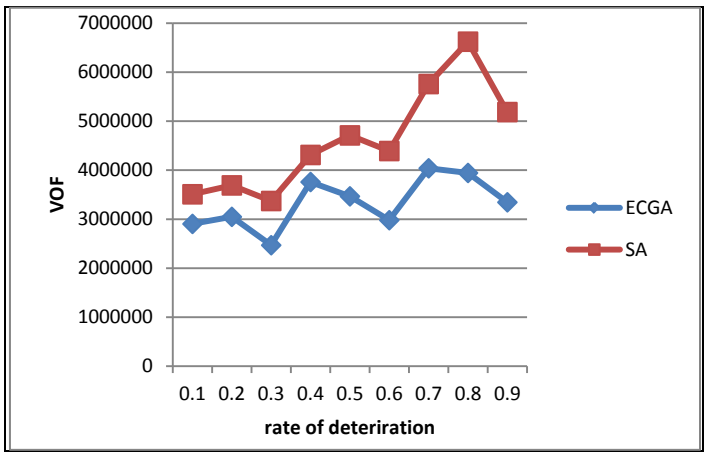

Fig. 1 Comparison of Sensitiveness to Deterioration Rate

As shown in figure 1, the problem has not any sensitivity to deterioration rate and for all the instances the ECGA algorithm yields better solution.

5. CONCLUSION

In this paper, the problem of parallel machine scheduling problem was considered and a model was developed for it based on makespan function. In order to solve the proposed model two meta-heuristics involving ECGA and SA were used. In computational experiments section the problem was solved for various instances and the sensitivity analyze is implemented for some important factors of proposed model.

For future research, solution method can be improved by using other meta- heuristic methods. Developing the model by proposing multi objective functions and assuming more complicated maintenance policies can also be considered as future research area.

REFERENCES

[1] C.N. Potts, M.Y. Kovalyov,(2000) "Scheduling with batching: A review," European Journal of Operational Research, Vol. 120. pp. 228–249.

[2] C.S. Sung, Y.H. Kim, S.H. Yoon,(2000) " A problem reduction and decomposition approach for scheduling for a flow shop of batch processing machines," European Journal of Operational Research, Vol. 121, pp. 179-192.

[3] C.S. Sung, S.H. Yoon,(1997) "Minimizing maximum completion time in a two-batch-processing-machine flowshop with dynamic arrivals allowed," Engineering Optimization, Vol. 28, pp. 231-243.

[4] C.S.Sung, Y.I.Choung,(2000) "Minimizing makespan on a single burn-in oven in semiconductor manufacturing," European Journal of Operational Research, Vol. 120, pp. 559–574.

[5] F.J. Ghazvini, L. Dupont,(1998) " Minimizing mean flow times criteria on a single batch processing machine with non-identical job sizes," International Journal Production Economics, Vol. 55, pp. 273–280.

[6] Lee CY, Uzsoy(1999) R. Minimizing makespan processing machine with dynamic job arrivals. International on a single batch Journal of Production Research;37:219–36.Liu ZH, Yu WC. Scheduling one batch processor subject.

[7] IKURA, Y. and GIMPLE, M., (1986), Scheduling algorithms for a single batch processing machine. Operations Research Letters, 5(2), 61–65.

[8] CHANDRU, V., LEE, C. Y. and UZSOY, R., (1993a), Minimizing total completion time on batch processing machines. International Journal of Production Research, 31(9), 2097–2121.

[9] DOBSON, G. and NAMBIMADOM, R. S.,(1992), The batch loading and scheduling problem.Research Report, Simon School of Business Administration, University of Rochester, Rochester, NY.

[10] DOBSON, G. and NAMBIMADOM, R. S., (2001), The batch loading and scheduling problem. Operations Research, 49(1), 52–65.

[11] UZSOY, R., (1994), Scheduling a single batch processing machine with non-identical job sizes. International Journal of Production Research, 32(7), 1615–1635.

[12] Uzsoy R.(1995) Scheduling batch processing machines with incompatible job families. International Journal of Production Research; 33: 2685–708.

[13] KEMPF, K. G., UZSOY, R. and WANG, C. S., (1998), Scheduling a single batch processing machine with secondary resource constraints. Journal of Manufacturing Systems, 17(1), 37–51.

[14] Dupont, L., Dhaenens-Flipo, C.,(2002). Minimizing the makespan on a batch processing machine with non-identical job sizes: an exact procedure. Computers & Operations Research 29, 807–819.

[15] Melouk, S., Damodaran, P., Chang, P.-Y., (2004). Minimizing makespan for single machine batch processing with nonidentical job sizes using simulated annealing. International Journal of Production Economics 87, 141–14.

[16] Damodaran P, Manjeshwar PK, Srihari K.(2006) Minimizing makespan on a batch- processing machine with non-identical job sizes using genetic algorithms. International Journal of Production Economics;103(2):882–91.

[17] Kashan A, Karimi B, Jolai F.(2006) Effective hybrid genetic algorithm for minimizing makespan on a single-batch-processing machine with non-identical job sizes. International Journal of Production Research;44(12):2337–60.

[18] CHANDRU, V., LEE, C. Y. and UZSOY, R.,(1993a), Minimizing total completion time on batch processing machines. International Journal of Production Research, 31(9), 2097–2121.

[19] UZSOY, R., (1995), Scheduling batch processing machines with incompatible job families. International Journal of Production Research, 33(10), 2685–2708.

[20] HOCHBAUM, D. S. and LANDY, D.,(1997), Scheduling semiconductor burn-in operations to minimize total flow time. Operations Research, 45(6), 874–885.

[21] CHANDRA, P. and GUPTA, S., (1997), Managing batch processors to reduce lead time in a semiconductor packaging line. International Journal of Production Research, 35(3), 611–633.

[22] Koh S-G, Koo P-H, Ha J-W, Lee W-S.(2004) Scheduling parallel batch processing machines with arbitrary job sizes and incompatible job families. International Journal of Production Research;42:4091–107.

[23] Chang, P., Damodaran, P., &Melouk, S. (2004). Minimizing makespan on parallel batch processing machines. International Journal of Production Research, 42, 4211–4220

[24] Lin BMT, Jeng AAK.(2004) Parallel-machine batch scheduling to minimize the maximum lateness and the number tardy jobs. International Journal of Production Economics;91(2):121–34

[25] Malve S, Uzsoy R.(2007) A genetic algorithm for minimizing maximum lateness on parallel identical batch processing machines with dynamic job arrivals and incompatible job families. Computers & Operations Research;34(10): 3016–28.

[26] Chung, S. H., Tai, Y. T., &Pearn, W. L. (2008). Minimisingmakespan on parallel batch processing machines with non-identical ready time and arbitrary job sizes. International Journal of Production Research, 47, 5109–5128

[27] Damodaran, P., Velez-Gallego, M. C., & Maya, J. (2009). A GRASP approach for makespan minimization on parallel batch processing machines. Journal of Intelligent Manufacturing. doi:10.1007/s10845-009-0272-z.

[28] Ahmadi JH, Ahmadi RH, Dasu S, Tang CS.(1992) Batching and scheduling jobs on batch and discrete processors. Operations Research; 39: 750–63.

[29] Sung CS, Kim YH(2003). Minimizing due date related performance measures on two batch processing machines. European Journal of Operational Research;147(3):644–56

[30] Mosheiov G, Oron D, RitovY(2004). Flow-shop batch scheduling with identical processing-time jobs. Naval Research Logistics;51(6):783–99.

[31] Lin BMT, Cheng TCE, Chou ASC (2007). Scheduling in an assembly type production chain with batch transfer. Omega;35: 143–51.