A Genetic Algorithm to Minimize Makespan and Number of Tardy Jobs in Parallel Machine Scheduling Problems

Ural Gökay ÇİÇEKLİ

Ege Üniversitesi, İİBF, İşletme Bölümü, Türkiye
gokaycicekli@gmail.com
(Geliş/Received: 17.12.2015; Kabul/Accepted: 20.05.2016)
DOI: 10.17671/btd.99806

***Abstract*—** This paper presents a genetic algorithm solution for the parallel machine scheduling problems with a real factory case. Various genetic components and operators have been examined to design a genetic algorithm for a parallel machine scheduling problem with an objective of minimizing the makespan and the number of tardy jobs. A production schedule has been optimized by using a genetic algorithm and results have been compared. The experimental results demonstrates that a genetic algorithm encoding method is performed successfully to achieve a solution for parallel machine problems.

***Keywords*—** Genetic algorithm, Parallel machine scheduling, tardy jobs

1. INTRODUCTION

Scheduling plays an important role in the performance of a manufacturing system and its efficiency. Scheduling is a decision-making process that deals with the allocation of resources to tasks over given time periods and its goals are to optimize one or more objectives [1]. Scheduling is the last stage of planning before production occurs [2]. Production scheduling can be defined as the allocation of available production resources over time to best satisfy some set of criteria [3]. Machine scheduling problems are grouped into several classes by changing the machine and process types and numbers in the system, and parallel machine scheduling is one of these classes.

A parallel machine environment is very common in the manufacturing industry. In the classical parallel machine scheduling problem, there are a sequence of n jobs with processing times $\{p_1, p_2,\ldots, p_n\}$ to be processed on m parallel machines $\{M_1,M_2, \ldots,M_m\}$ [4]. The parallel machine scheduling (PMS) problem involves two kinds of decisions, sequencing and job-machine assignment [5]. Any job can run on any machine, each having a different processing time.

In today's industries, customer satisfaction is imperative in manufacturing and meeting due dates are very important to sustain competitive strength. As such, scheduling jobs on parallel machines against their due dates has become a very common setting from a practical perspective [6]. It is extremely important to complete jobs no later than their due dates thru proper scheduling. Minimizing the number of tardy jobs criterion is equivalent to maximizing the average number of early jobs criterion [7]. As such, the number of tardy jobs has become increasingly important in scheduling.

In real life applications, scheduling problems involves solving for an optimal schedule under various objectives, different machine environments, and with a consideration for unique job characteristics [8]. Scheduling problems can have a different level of complexity. This complexity makes it challenging to find an optimal solution. Many methods are used to solve these difficult scheduling problems. One approach is the use of a genetic algorithm (GA). GAs became well known due to their efficiency in solving combinatorial optimization problems such as scheduling [9].

GA is a type of global optimization and a stochastic search technique that is based on the mechanism of natural evolution. GA can rapidly find a solution that is close to optimum in difficult and huge search spaces. Today's competitive business environment has led companies to look for rapid and acceptable solutions [10].

GA is also good at solving continuous and discrete combinatorial problems. A GA does not focus on sub-optimization, and instead it evaluates to find the optimum value for the objective function. Instead of ensuring an optimal solution each time, genetic algorithm solutions are very close and slightly less efficient to optimum [11].

As such, it is more important to select a proper method, representative coding and to define the objective for a GA in complex problems. GAs deal with a parameter code, not the parameters themselves [12]. As such, a GA is applicable to problems having discontinuous functions [11]. GAs carry the best individual (having the best objective function value) to the next generation. It is well known that the problem of minimizing makespan and total tardiness on parallel machines is NP-hard (non-deterministic polynomial-time hard) [13][14][15][16].

2. PARALLEL MACHINE SCHEDULING BY GENETIC ALGORITHMS

Scheduling problems form an important class of combinatorial optimization problems. They are typically combinatorial problems that cannot be formulated as linear programs and there are no simple rules or algorithms that yield optimal solutions in a limited amount of computer time [17]. It is difficult to find a solution to combinatorial optimization problems by deterministic methods. GA can be used to solve combinatorial optimization problems [18].

GA is a type of global optimization and a stochastic search technique that is based on the mechanism of natural evolution. GA was first introduced by John Holland in 1960. Genetic algorithms are search methods based on the genetic processes in the nature. GA is a random search technique that aims to find global optimization in a complex search space [19].

The starting point in constructing the GA is to define an appropriate genetic encoding. Defining proper encoding is crucial because it significantly affects all the subsequent steps of the GA [20]. Thus, it is important to find a fit encoding method such as binary encoding, permutation encoding, value encoding and tree encoding, before GA is applied to a problem. Permutation coding is the best method for ordering problems [21]. Thus, permutation coding is used for scheduling problems.

During the evolution of the population, operators of a genetic algorithm play a significant role. These operators are selection, crossover, and mutation. Solutions are encoded in a form called a chromosome. Each chromosome shows a complete solution to a problem. Each chromosome has an objective function value that is used for carry the best individual to the next generation.Parallel machine scheduling problems are aimed at minimizing makespan. Each machine's duration times are calculated, and maximum duration time is total duration time. This is an objective function value for parallel machine scheduling that is free from setup times.

GA is widely used for solving machine scheduling problems. Many studies have illustrated that GA performs well for solving a parallel machine scheduling problem [22][23][24]. A simple example is 3 machines and 5 jobs as a given representative process for genetic algorithm addressing parallel machine scheduling. Every machine has a different completion for each job. The chromosome can be represented as follows for a schedule as shown in Table 1.

Table 1. An example schedule for 3 machines and 5 jobs schedule.

Machine 1	*Job 5*	
Machine 2	*Job 4*	*Job 1*
Machine 3	*Job 2*	*Job 3*

Machine 1					**Machine 2**					**Machine 3**				
Job 5	*null*	*Null*	*null*	*null*	**Job 4**	**Job 1**	*null*	*null*	*null*	**Job 2**	**Job 3**	*null*	*null*	*null*

There are many types of genetic crossover operators for permutation encoding, such as position-based crossover, order based crossover, one point crossover, two point crossover, cycle crossover, order crossover, linear order crossover, and partially mapped crossover. A position-based crossover operator was used in this study.

A position-based crossover needs a random selected parent string to decide which genes (dominant gene) must be selected for offspring. A set of the first offspring's positions from the first parent is selected when a parent string value is "1". Each position of a gene is independently marked with a probability of 0.5. After that, to fill empty genes, the second parent's genes are taken left from right in order without any repetition. The same procedure is applied for the second offspring, but the procedure then begins with parent two, while the dominant parent string value is "0". This is shown in Figure 1. This is an example of two parents and a random parent string that generates two offspring without any mutation by using a permutation GA.Production schedules for generated offspring 1 and offspring 2 from the parent 1 and parent 2 are shown in Table 2.

Parent 1	**Job 5**	*101*	*102*	*103*	*104*	**Job 4**	**Job 1**	*105*	*106*	*107*	**Job 2**	**Job 3**	*108*	*109*	*110*
Parent 2	**Job 4**	**Job 5**	**Job 2**	*101*	*102*	**Job 3**	*103*	*104*	*105*	*106*	**Job 1**	*107*	*108*	*109*	*110*
Parent String	1	1	0	0	1	0	0	1	0	1	0	0	1	1	1
Offspring 1	**Job 5**	*101*	**Job4**	**Job2**	*104*	*102*	**Job 3**	*105*	*103*	*107*	*106*	**Job 1**	*108*	*109*	*110*
Offspring 2	**Job 5**	*102*	**Job 2**	*101*	*104*	**Job 3**	*103*	**Job 4**	*105*	*106*	**Job 1**	*107*	*108*	*109*	*110*

Figure 1. One iteration for two parents and a random parent string generates two offspring by using Position Based Crossover (PBX).

Table 2. Production schedules for generated offspring 1 and offspring 2 from the parent 1 and parent 2.

Offspring 1	**Machine 1**	*Job 5*	*Job 4*	*Job 2*
	Machine 2	*Job 3*		
	Machine 3	*Job 1*		
Offspring 2	**Machine 1**	*Job 5*	*Job 2*	
	Machine 2	*Job 3*	*Job 4*	
	Machine 3	*Job 1*		

3. FINDINGS

In this study, I developed an algorithm with an encoding method written by Kocamaz et al.[25]. I added minimizing total tardiness to their algorithm. The code for the algorithm was written by using VBA (Visual Basic for Application) with a Microsoft Excel interface. This developed algorithm was applied to a real case as an experimental study.

I made some general assumptions about the scheduling:

- All jobs and machines are available at time Zero.
- Machines never break down and are available throughout the scheduling period.
- Pre-emption is not allowed.
- Machines may be idle.
- Splitting of job or job cancellation is not allowed.

In order to determine of the performance level of the algorithm, the number of iterations and optimization times were calculated for 216 jobs over 25 times. These jobs were completed by three parallel machines. Every machine isn't capable of finishing all jobs and every machine has different competition times for the same job. Sample production times are given in Table 3. An "X" means that a machine isn't capable of finishing a job.

I considered setup times that are needed between processing of jobs for preparation of the production line for an coming job. Jobs are partitioned into families and setup time is required between these families. Therefore, there are different setup requirements in different jobs for it to be ready to start the next job. Additionally, setup time doesn't occur between the same family's jobs.

In Figure 2, the number of succeed due date are given for a best solution and the average of 25 repetitions with each iteration. Series one represents the number of succeed due date for an average. A solution was developed from 157 to 181.43. In series 2, the number of succeed due date for best solution was found. The values are 157 for the 1st iteration and 185 for the 10,000th iteration.

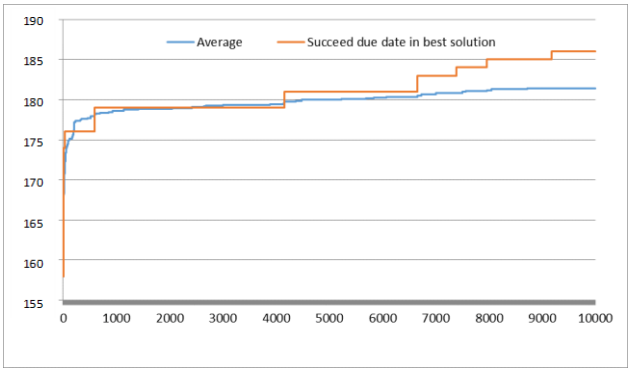

Figure 2. Number of Succeed Due Date

Figure 3. Shows makespan time for best makespan solution and average repetition. Makespan decreases through the last iteration. The value is of the best makespan because the algorithm tries to improve upon multiple-criteria for scheduling.

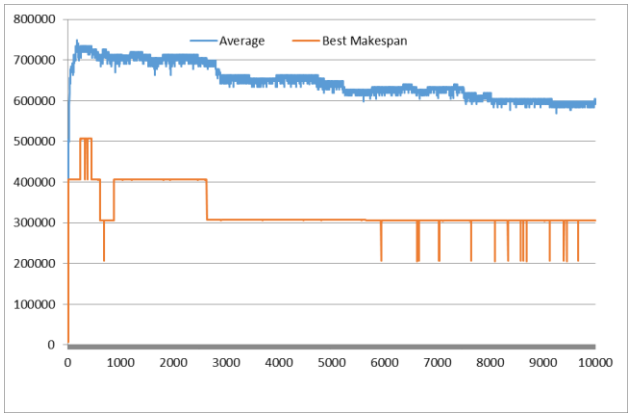

Figure 3. Makespan times for average and best makespan solution

Table 3. Sample production times for each machine.

	Job 1	**Job 2**	**Job 3**	**Job 4**	**Job 5**	**Job 6**	**Job 7**	**Job 8**	**Job 9**	**Job 10**	
Machine 1	59	15	20	12	23	15	18	12	7	5	…
Machine 2	94	24	20	12	25	18	20	13	X	X	…
Machine 3	X	X	X	X	X	X	X	X	X	X	…

Iteration number of achieved solutions are represented in Figure 4. The series represents respectively the iteration level of achieved due date, best solution and makespan.

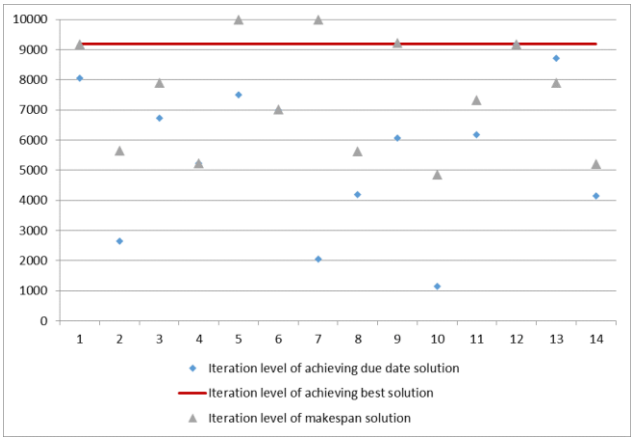

Figure 4. Solution iteration level

I started with an initial parent by human decision. This resulted in 157 achieved due dates and 406,750 minutes for makespan. After 10,000 iterations, the best solution was 185 achieved due dates and 605,265 minutes for the makespan. The schedule plan for the best solution is given in Table 4, Table 5, Table 6. Underlined jobs represent jobs completed past their due dates.

Table 4. Machine I schedule plan

Seq.	Jobs	Seq.	Jobs	Seq.	Jobs
1	1	26	71	51	129
2	2	27	37	52	149
3	3	28	76	53	153
4	4	29	53	54	151
5	5	30	54	55	117
6	6	31	84	56	157
7	7	32	77	57	147
8	9	33	81	58	158
9	10	34	75	59	167
10	11	35	78	60	161
11	16	36	79	61	214
12	8	37	101	62	146
13	29	38	118	63	155
14	22	39	86	64	154
15	12	40	87	65	127
16	30	41	113	66	51
17	31	42	88	67	152
18	33	43	121	68	189
19	34	44	115	69	145
20	35	45	150	70	196
21	32	46	123	71	114
22	46	47	126	72	64
23	49	48	82	73	48
24	50	49	120	74	58
25	124	50	122	75	80

Table 5. Machine II schedule plan

Seq.	Jobs	Seq.	Jobs	Seq.	Jobs
1	59	26	106	51	187
2	41	27	107	52	173
3	26	28	72	53	163
4	170	29	89	54	166
5	65	30	73	55	204
6	20	31	111	56	159
7	96	32	109	57	171
8	25	33	112	58	61
9	21	34	135	59	168
10	94	35	103	60	188
11	125	36	136	61	172
12	38	37	165	62	174
13	70	38	160	63	119
14	46	39	98	64	110
15	39	40	164	65	60
16	68	41	55	66	14
17	191	42	130	67	99
18	56	43	203	68	133
19	40	44	142	69	116
20	95	45	141	70	85
21	93	46	162	71	176
22	57	47	175	72	15
23	105	48	156	73	74
24	104	49	199	74	47
25	91	50	148		

Table 6. Machine III schedule plan

Seq.	Jobs	Seq.	Jobs	Seq.	Jobs
1	92	26	184	51	24
2	13	27	183	52	200
3	218	28	202	53	144
4	28	29	182	54	186
5	215	30	138	55	198
6	17	31	208	56	213
7	67	32	137	57	197
8	100	33	194	58	139
9	18	34	181	59	128
10	52	35	185	60	27
11	178	36	209	61	69
12	23	37	190	62	143
13	102	38	97	63	108
14	132	39	207	64	36
15	43	40	195	65	66
16	134	41	212	66	216
17	62	42	210	67	169
18	180	43	201	68	44
19	131	44	179	69	83
20	19	45	211		
21	42	46	90		
22	63	47	205		
23	217	48	177		
24	192	49	206		
25	140	50	193		

4. CONCLUSION

In manufacturing today, it is vital to satisfy customer demands in a timely manner. Therefore, minimizing total tardiness isa significant issue. Further, makespan is important to maintain production efficiency. In this study, a parallel machine setup was optimized using a genetic algorithm approach. The results minimized makespan and the number of tardy jobs in a parallel machine setup. The experiment results demonstrated that the genetic algorithm performs successfully to achieve an optimal solution. Looking ahead, our future research will investigate uncertainties in scheduling when minimizing makespan, total tardiness, total work-in process and stock cost of finished goods in a parallel machine setup.

REFERENCES

[1] M. Pinedo, **Scheduling: Theory, Algorithms, and Systems**, Edition: 3, Springer, 2008.

[2] R. S. Russell, B. W.Taylor III, **Operation Management**, 4th Edition, Prentice Hall, New Jersey, 2000.

[3] S. C. Graves, "A Review of Production Scheduling", *Operations Research,* 29(4), 646-675,1981.

[4] W. Zhenbo, X. Wenxun, "Parallel Machine Scheduling with Special Jobs", *Tsınghua Science and Technology*, 11(1), 107-110, 2006.

[5] E. Mokotoff, "Parallel machine scheduling problems: A survey", *Asia - Pacific Journal of Operational Research*, 18, 193-242, 2001.

[6] D. Biskup, J. Herrmann, J. N. D. Gupta, "Scheduling identical parallel machines to minimize total tardiness",*International Journal of Production Economics*, 115(1), 134-142, 2008.

[7] E. O. Oyetunji, "Some Common Performance Measures in Scheduling Problems: Review Article", *Research Journal of Applied Sciences, Engineering and Technology*, 1(2), 6-9, 2009.

[8] P. Pandian, P. Rajendran, "Solving Constrained Flow-Shop Scheduling Problems with Three Machines", *International Journal of Contemporary Mathematical Sciences*, 5(19), 921-929, 2010.

[9] H. Boukef, M. Benrejeb, P. Borne, "A Proposed Genetic Algorithm Coding for Flow-Shop Scheduling Problems", *International Journal of Computers, Communications & Control*, II(3), 229-240, 2007.

[10] M. Kocamaz, U. G. Çiçekli, "Paralel Makinaların Genetik Algoritma İle Çizelgelenmesinde Mutasyon Oranının Etkinliği", *Ege Academic Review*, 10(1), 199-210, 2010.

[11] M. F. Yeo, E. O. Agyei (1998): "Optimising engineering problems using genetic algorithms", *Engineering Computations*, 15(2), 268-280.

[12] Y. Z. Wang, "Using genetic algorithm methods to solve course scheduling problems",*Expert Systems with Applications*, 25, 39-50, 2003.

[13] M. R. Garey, D. S. Johnson, "Strong NP-completeness Results: Motivation, Examples and Implications", *Journal of the Association for Computing Machinery*, 25, 499-508, 1978.

[14] J. C. Ho, Y. L. Chang, "Heuristics for minimizing mean tardiness for m parallel machines", *Naval Research Logistics*, 38, 367-381, 1991.

[15] R. M. Karp,"Reducibility among combinatorial problems: complexity of computer computations", New York: Plenum Press, 85–103, 1972.

[16] C. Koulamas, "Decomposition and hybrid simulated annealing heuristics for the parallelmachine total tardiness problem", *Naval Research Logistics*, 44, 109–125, 1997.

[17] M.J. Tarokh, M. Yazdani, M. Sharif, M. N. Mokhtarian, "Hybrid Meta-heuristic Algorithm for Task Assignment Problem", *Journal of Optimization in Industrial Engineering*, 7, 45-55, 2011.

[18] M. Mitchell, **An Introduction to Genetic Algorithms**, MIT Press, 1998.

[19] M. Mori, C. C Tseng, "Theory and Methodology: A genetic algorithm for multi-mode resource constrained project scheduling problem",*European Journal of Operational Research*,1 00, 134-141, 1997.

[20] Z.X. Guo, W.K.Wong,S.Y.S.Leung, J.T.Fan, S.F.Chan, "Genetic optimization of order scheduling with multiple uncertainties", *Expert Systems with Applications*, 35, 1788-1801, 2008.

[21] P. Borovska, "Solving the Travelling Salesman Problem in Parallel by Genetic Algorithm on Multicomputer Cluster", **International Conference on Computer Systems and Technologies - CompSysTech'06**, University of Veliko Tarnovo, Bulgaria,15-16 June 2006.

[22] A. H.Abdekhodaee, A. Wirth, H.S. Gana, "Scheduling two parallel machines with a single server: the general case", *Computers & Operations Research*, 33,994-1009, 2006.

[23] F. S. Serifoglu, G. Ulusoy, "Parallel machine scheduling with earliness and tardiness penalties", *Computers & Operations Research*, 26, 773-787, 1999.

[24] F. Yalaoui, C. Chu, "Parallel machine scheduling to minimize total tardiness", *International Journal of Production Economics*, 76(3), 265-279, 2002.

[25] M. Kocamaz, U. G. Çiçekli, H. Soyuer, "A Developed Encoding Method for Parallel Machine Scheduling with Permutation Genetic Algorithm", **European and Mediterranean Conference on Information Systems EMCIS 2009**, Crowne Plaza Hotel, Izmir, Brunel University and Dokuz Eylül University, 13 – 14 July 2009.