Research Journal of Applied Sciences, Engineering and Technology 11(6): 666-673, 2015
DOI: 10.19026/rjaset.11.2029
ISSN: 2040-7459; e-ISSN: 2040-7467

Submitted: May 30, 2015 Accepted: July 8, 2015 Published: October 25, 2015

Research Article

An Evolutionary Algorithmic Approach for Single Machine Early Tardy Scheduling Problem

R. Jayabhaduri
Department of Computer Science and Engineering, Sri Venkateswara College of Engineering, Irungattukottai-602 117, Chennai, Tamilnadu, India

Abstract: Most of the real world scheduling problems incorporates Just-In-Time production philosophy which leads to a growing interest in the development of various nature inspired metaheuristic algorithms. Single Machine Early Tardy scheduling problem (SMETP) is one such problem in which jobs have to be scheduled on a single machine against a restrictive common due date parameter and this problem is strongly a NP-hard combinatorial optimization problem. As job sizes vary from 10 to 1000, problems of larger job sizes cannot be solved by exact algorithms. Hence in this research study, we propose genetic algorithm with variations in local search to find an optimal schedule which jointly minimizes the summation of earliness and tardiness cost penalties of 'n' jobs from a common due date by satisfying the three SMETP scheduling properties. The performance of this evolutionary algorithm is validated on the 280 benchmark instances proposed by Biskup and Feldmann for various job sizes and the results show that genetic algorithm works well for smaller job sizes.

Keywords: Common due date, genetic algorithm, heuristics, local search, single machine scheduling

INTRODUCTION

Sequencing and scheduling are decision making processes which play a crucial role in manufacturing and production industries. Scheduling jobs on a Single machine against a restrictive common due date to Minimize Early and Tardy Penalties (SMETP) has been studied by many researchers for nearly 3 decades with the principles of Just-In-Time (JIT) inventory management. SMETP problem belongs to a class of scheduling problems formally defined as $1/d/\sum_{i=1}^{n} \alpha_i E_i + \beta_i T_i$. Each job has three characteristics namely processing time, earliness penalty cost and tardiness penalty cost associated with it. In this study, we focus on single machine restricted common due date problems where some jobs may be completed before the common due date which is referred to as earliness penalty and after the common due date is referred to as tardiness penalty respectively. Execution of jobs before the common due date may lead to storage of products in the industries, whereas execution of jobs after the common due date may lead to loss of reputation of customer's goodwill. Hence the objective is to find an optimal schedule which exactly finishes execution of jobs on the common due date to jointly minimize the summation of earliness and tardiness penalties costs.

Common due date scheduling problems are categorized into restrictive and unrestrictive ones. In the case of unrestricted common due date scheduling problems, all jobs complete its execution before the common due date. Hence there is no challenge in optimizing algorithms. In our research study, we focus on restrictive single machine common due date scheduling problems.

In this study, we address the issues of minimizing error offset for the evolutionary algorithms, applying local improvement methods in the algorithms to reduce the computation time and when to terminate the algorithm.

Problem formulation and properties of SMETP: Restrictive common due date scheduling problem is formulated as follows:

- 'n' jobs are available for processing at time zero, which have to be processed on a single machine.
- Each of the jobs needs exactly one operation. No pre-emption of jobs is allowed.
- The processing times p_j of the jobs 1...n are deterministic, common due date 'd' is computed as:

  $$d = \sum_{i=1}^{n} p_j * h$$

  where h is the restrictive common due date parameter.
- Completion time C_j of each job is computed by:

  $$\sum_{i=1}^{n} . p_{j-1} + p_j$$

- A job is referred to as early if its completion time falls below the common due date and the earliness penalty of job j is given by $E_j = \max(0, d - C_j)$.
- A job is tardy if its processing time ends after common due date and the tardiness penalty is computed by $T_j = \max(C_j - d, 0)$ respectively for all jobs j = 1..n.

The objective in our study is to find an optimal schedule σ which jointly minimizes the earliness and tardiness penalties of all jobs closer to the due date is given as:

$$\sigma = \sum_{i=1}^{n} \propto iEi + \beta i\ Ti$$

For the restricted SMETP, an optimal schedule should satisfy the following three optimality properties.

Property 1: No idle times are inserted between consecutive jobs (Cheng and Kahlbacher, 1991).

Property 2: The optimal schedule is 'V' shaped around the common due date. But a straddling job may exist, i.e., a job whose execution starts before and finishes after the due date, (Baker and Scudder, 1989).

Property 3: Either the processing time of first job starts at time zero or one job is completed at the due date in the optimal schedule (Hoogeveen and van de Velde, 1991).

Due to the complexity of SMETP, branch and bound techniques and several nature inspired metaheuristic algorithms like genetic algorithms, simulated annealing, tabu search, differential evolution, artificial bee colony optimization and hybrid evolutionary algorithms are addressed by various researchers to tackle this problem.

LITERATURE REVIEW

Many researchers have studied about the nature of SMETP which is strongly proved to be a NP hard combinatorial optimization problem. Baker and Scudder review the literature on scheduling models with Early Tardy penalties in 1989. The researchers pointed out that the single-machine scheduling problem with a restricted common due date has never been addressed in the literature. By that time, Hall *et al*. (1991) proved that this problem is NP-hard. Due to its complexity, many authors addressed this problem using nature inspired metaheuristic methods and compared their results with state-of-the-art metaheuristics. Lee and Kim (1995) developed a parallel genetic algorithm, while James (1997) used Tabu search approach to address this.

Biskup and Feldmann (2001) presented 280 benchmarks for the restrictive common due-date problem with general earliness and tardiness penalties. Feldmann and Biskup (2003) studied the restricted E/T problem postponing the schedule by applying different metaheuristics: Evolutionary Search (ES), Simulated Annealing (SA) and Threshold Accepting (TA). Lin *et al*. (2007) used a sequential exchange approach while Liao and Cheng (2007) proposed a variable neighborhood search for minimizing single machine weighted earliness and tardiness with common due date. Nearchou (2008) used Differential Evolution algorithm. Le and Hong (2013) developed a hybrid metaheuristic Permutation-based Harmony search algorithm by incorporating Variable Neighborhood Search (PHVNS) and demonstrated that their algorithm shows high competitiveness by comparing with some state-of-the-art metaheuristics.

MATERIALS AND METHODS

Proposed algorithm:

Genetic algorithm with local improvement for SMETP: John Holland's invention of Genetic algorithm is a population-based metaheuristic evolutionary algorithm that evolves from one population of chromosomes to a new population by natural evolution such as reproduction, crossover and mutation and follows Charles Darwin's "Survival of the fittest".

Sequence representation: We use permutation encoding mechanism to represent a sequence of jobs. A sequence is mapped into a chromosome with the alleles assuming different and non-negative integer values in the (1..n) interval. For a 5 jobs problem, the complete sequence is represented as ((1) (2) (3) (4) (5)) where [i] is the position of the i^{th} job in the sequence. The objective function is to find a sequence σ which jointly minimizes the sum of early and tardy cost penalties for single machine restrictive common due date scheduling problems.

Initial population generation and fitness evaluation: Jobs are scheduled according to p/α heuristic for jobs that complete before the common due date and p/β heuristic for jobs that complete after the common due date respectively. This is used to generate the first individual in the initial population. The remaining sequences in initial population are generated by constructive heuristics which places [i] job in all possible combinations.

Reproduction of chromosomes: We have used roulette wheel selection strategy to select the chromosome with minimum fitness value to evolve from current population to the new population.

Ordered crossover: We have implemented ordered crossover operator for the mating parents which are selected randomly from the mating pool. Two crossover points for the mating parents are randomly generated to determine the range for crossover. The length of the crossover is in the range with Lower Limit (LL) (1, n-1)

P_1	1	2	3	4	5	6	7	8	9	10
P_2	3	6	1	10	8	4	9	7	2	5

Fig. 1: Ordered crossover operation

P_1	1	2	3	4	5	6	7	8	9	10
	Position 1: 3									
	Position 2: 9									
O_1	1	2	9	3	4	5	6	7	8	10

Fig. 2: Sliding mutation operation

P_1	1	2	3	4	5	6	7	8	9	10
Allele 1: 3, Allele 2: 9										
O_1	1	2	9	4	5	6	7	8	3	10

Fig. 3: Pair-wise random swap mutation operation

P_1	1	2	3	4	5	6	7	8	9	10
Allele 1: 3, Allele 2: 4										
O_1	1	2	4	3	5	6	7	8	9	10

Fig. 4: Adjacent pair-wise swap mutation operation

job position and the Upper Limit (UL) (LL, n). Ordered crossover is explained with an example (Fig. 1):

LL = 3, UL = 7

O_1 5 6 1 10 8 4 9 7 2 3

Offspring is generated by retaining the elements of the parent that falls within the crossover range and inheriting the remaining elements from the parent in the order in which they appear in that parent beginning from the first position following the second crossover point and the elements are skipped if they are already present in the newly generated offspring.

Mutation: The resultant offspring represents the sequence σ and the value of the total early and tardy penalties z (σ) is calculated by using Eq. (1). We have implemented sliding mutation strategy followed by pair-wise random swap mutation. Two alleles in a parent are selected based on two randomly generated positions. The allele in position 2 is shifted to the allele in position 1 and the allele in position 1 is shifted right by 1 place and follows the sane order of alleles in the parent and results in new offspring (Fig. 2).

The newly generated offspring again undergoes pair-wise random swap mutation and generates a new offspring σ1. Two alleles in a parent are selected randomly and their positions are swapped and result in a new offspring (Fig. 3).

The fitness function z (σ1) is computed and checked with the fitness value of z (σ). If z (σ1) < z (σ), the newly generated offspring is added to the population set and the sequence with fitness value z (σ) is removed from the population set by applying elitism replacement strategy. If there is no improvement after several generations, the original offspring is added to the population set.

Mutation with local improvement: Each resultant offspring of job size 'n' generated after mutation operation again undergoes adjacent pair-wise swap mutation and yields new sequences (Fig. 4).

The fitness function is computed for all new offsprings and the offspring which returns minimum fitness value is added to the new population set.

Finally out of n * n offsprings, 'n' offsprings are added to the new population set which form the chromosomes for the next generation. This undergoes roulette wheel selection, ordered crossover and mutation with local improvement for subsequent generations.

Algorithm 1:
Algorithm for initial population generation and fitness evaluation: Procedure Init_Population

Input: Number of instances, number of jobs, processing time for each operation, earliness penalty for each job, tardiness penalty for each job, common due date for the jobs

Output: Schedule of jobs
Encode each individual in population of size, indiv
Generate initial population by calling Init_Population() method.
While stopping criteria not met
1. Sort jobs according to Shortest Processing Time heuristic to construct the initial sequence.
2. Construction of remaining sequences by constructive heuristics.
   Repeat
3. Compute processing time, completion time, earliness and tardiness cost penalties of all jobs in the sequence as
   a. ctime [j] = ctime [j-1] + ptime [j]
   b. ptm + = ptime [j]
   c. early [j] = cdd-ctime [j]
   d. tardy [i] = ctime [j] -cdd
4. Place jobs to the left in 'V' shaped arrangement for the jobs with completion time less than the common due date value; otherwise place jobs to its right.
5. Compute fitness value of the sequence.
   Until the last sequence in initial population

End while
End procedure

Algorithm 2:
Genetic algorithm: Procedure ga ()

Input: Number of jobs, processing time for each operation, earliness penalty for each job, tardiness penalty for each job, common due date for the jobs

Output: Schedule of jobs
Perform Roulette wheel selection strategy
Do ordered crossover to generate a new offspring
Perform random swap and sliding mutation

Algorithm 3:
Local improvement: For all newly generated members in the population after mutation
1. Choose a newly generated offspring (σ2) of job size n
2. Generate new sequences for the offspring (σ2) by adjacent pair-wise swap mutation.
3. Evaluate the fitness function for the newly generated offsprings.
4. Retain the offspring with the minimum fitness value.
5. Add this offspring to the new population set

Termination criteria: We have generated $2n + n^2$ individuals for all job sizes 'n' in each generation and the best 3n individuals are added to the population set. We run our genetic algorithm by fixing the number of generations as 1000.

Materials: The benchmark instances of restricted single-machine common due date problems are proposed by Biskup and Feldmann (2001) on job sizes n = 10, n = 20, n = 50, n = 100, n = 200, n = 500 and n = 1000. The common due date d is calculated by d = round (SUM_P * h) where round (X) gives the biggest integer which is smaller than or equal to X; Sum_P denotes the sum of the processing times of the n jobs and the parameter h is used to calculate more or less restrictive common due dates. For the following 280 benchmarks we used h = 0.2, h = 0.4, h = 0.6 and h = 0.8. The instances are available at http:// people.brunel.ac.uk/~mastjjb/jeb/orlib/schinfo.html.

RESULTS AND DISCUSSION

Experiments conducted: The developed evolutionary algorithm is implemented in Java language on a computer with 2.27 GHz Intel (R) Core i5 CPU and 3 GB RAM with main memory, running Windows 8.1 operating system with Java NetBeans IDE 8.2. We solved 280 benchmark instances for the different sizes n = 10, n = 20, n = 50, n = 100, n = 200, n = 500 and n = 1000 with h = 0.2, h = 0.4, h = 0.6 and h = 0.8. For all the 280 benchmark problems, parameters and its values chosen for our study are listed in Table 1.

Case I: The benchmark instances considered from OR-Library by J E Beasley has to schedule 10 jobs in a single machine for the values of common restrictive due date parameter 'h' taking h = 0.2, 0.4, 0.6 and 0.8. * indicates optimal objective function values. In the Table 2, the attributes UB represents known Upperbound function value, $COST_{GA}$ represents the fitness cost value obtained by job scheduling. The deviation in cost percentage is computed as % error = ((COST GA-UB) /UB) *100 and the results are tabulated and given below.

Results in Table 3 shows that optimal objective function values are achieved for some instances for the value of h taking 0.6 and 0.8.

Table 1: GA parameters and values

Parameters	Values
Population size	Job size
Crossover rate p_c	0.890
Mutation rate p_m	0.005
Number of GA runs	10.000
Number of generations (termination criteria)	1000

Table 2: Job size = 10, h = 0.2 and 0.4

	N = 10, h = 0.2			N = 10, h = 0.4		
Instance	UB	$COST_{GA}$	Error (%)	UB	$COST_{GA}$	Error (%)
k = 1	1936	1936	0.00	1025	1025	0.00
k = 2	1042	1001	-3.93	615*	615*	0.00
k = 3	1586	1586	0.00	917	917	0.00
k = 4	2139	2139	0.00	1230	1180	-4.07
k = 5	1187	1149	-3.20	630	619	-1.75
k = 6	1521	1469	-3.42	908*	908*	0.00
k = 7	2170	2102	-3.13	1374*	1374*	0.00
k = 8	1720	1680	-2.33	1020	1003	-1.67
k = 9	1574	1574	0.00	876*	876*	0.00
k = 10	1869	1869	0.00	1136	1097	-3.43

Results show that optimal objective function value is obtained for h = 0.4 and indicated by *

Table 3: Job size = 10, h = 0.6 and 0.8

	N = 10, h = 0.6			N = 10, h = 0.8		
Instance	UB	$COST_{GA}$	Error (%)	UB	$COST_{GA}$	Error (%)
k = 1	841*	860	2.26	818*	1022	24.94
k = 2	615*	877	42.60	615*	1432	132.85
k = 3	793*	927	16.90	793*	1301	64.06
k = 4	815*	815*	0.00	803	952	18.56
k = 5	521*	521*	0.00	521*	820	57.39
k = 6	755*	770	1.99	755*	904	19.74
k = 7	1,101	1083	-1.63	1,083*	1083*	0.00
k = 8	610*	610*	0.00	540*	540*	0.00
k = 9	582*	582*	0.00	554*	596	7.58
k = 10	710	710	0.00	671*	822	22.50

Table 4: Job size = 20, h = 0.2 and 0.4

	N = 20, h = 0.2			N = 20, h = 0.4		
Instance	UB	$COST_{GA}$	Error (%)	UB	$COST_{GA}$	Error (%)
k = 1	4,431	4394	-0.84	3,066	3073	0.23
k = 2	8,567	8430	-1.60	4,897	4799	-2.00
k = 3	6,331	6146	-2.92	3,883	3838	-1.16
k = 4	9,478	9203	-2.90	5,122	5118	-0.08
k = 5	4,340	4164	-4.06	2,571	2495	-2.96
k = 6	6,766	6527	-3.53	3,601	3536	-1.81
k = 7	11,101	10349	-6.77	6,357	6180	-2.78
k = 8	4,203	3920	-6.73	2,151	2106	-2.09
k = 9	3,530	3414	-3.29	2,097	2078	-0.91
k = 10	5,545	4979	-10.21	3,192	2930	-8.21

Table 5: Job size = 20, h = 0.6 and 0.8

	N = 20, h = 0.6			N = 20, h = 0.8		
Instance	UB	$COST_{GA}$	Error (%)	UB	$COST_{GA}$	Error (%)
k = 1	2,986	3230	8.17	2,986	4798	60.68
k = 2	3,260	3206	-1.66	2,980	3417	14.66
k = 3	3,600	3845	6.81	3,600	5534	53.72
k = 4	3,336	3317	-0.57	3,040	3419	12.47
k = 5	2,206	2215	0.41	2,206	3049	38.21
k = 6	3,016	3107	3.02	3,016	4859	61.11
k = 7	4,175	4131	-1.05	3,900	4368	12.00
k = 8	1,638	1704	4.03	1,638	2118	29.30
k = 9	1,992	2069	3.87	1,992	2819	41.52
k = 10	2,116	2091	-1.18	1,995	2669	33.78

Table 6: Job size = 50, h = 0.2 and 0.4

	N = 50, h = 0.2			N = 50, h = 0.4		
Instance	UB	$COST_{GA}$	Error (%)	UB	$COST_{GA}$	Error (%)
k = 1	42,363	40586	-4.19	24,868	23812	-4.25
k = 2	33,637	30661	-8.85	19,279	17907	-7.12
k = 3	37,641	34510	-8.32	21,353	20577	-3.63
k = 4	30,166	27691	-8.20	17,495	16794	-4.01
k = 5	32,604	32377	-0.70	18,441	18010	-2.34
k = 6	36,920	34893	-5.49	21,497	20517	-4.56
k = 7	44,277	42970	-2.95	23,883	23114	-3.22
k = 8	46,065	43761	-5.00	25,402	24978	-1.67
k = 9	36,397	34381	-5.54	21,929	19997	-8.81
k = 10	35,797	33080	-7.59	20,048	19311	-3.68

Case II: The benchmark instances considered have to schedule 20 jobs in a single machine for the values of common restrictive due date parameter 'h' taking h = 0.2, 0.4, 0.6 and 0.8, respectively.

Results shown in Table 4 prove that proposed algorithm has minimized the objective function fitness value very well than the upper bound for most of the instances (Table 5).

Case III: The benchmark instances considered have to schedule 50 jobs in a single machine for the values of common restrictive due date parameter 'h' taking h = 0.2, 0.4, 0.6 and 0.8, respectively.

The proposed algorithm generated better results for 50 jobs and the results are listed in Table 6 and 7.

Case IV: The benchmark instances considered have to schedule 100 jobs in a single machine for the values of common restrictive due date parameter 'h' taking h = 0.2, 0.4, 0.6 and 0.8, respectively.

The proposed algorithm has minimized the early tardy penalty costs for all instances for h = 0.2 and 0.4 and shown in Table 8 while Table 9 shows % error deviation to be high for h = 0.8.

Case V: The benchmark instances considered have to schedule 200 jobs in a single machine for the values of common restrictive due date parameter 'h' taking h = 0.2, 0.4, 0.6 and 0.8, respectively.

The proposed algorithm has minimized the early tardy penalty costs for all instances for h = 0.2 and some instances for h = 0.4 and the results are tabulated in Table 10.

Results shown in Table 11 shows that % error deviation is high for larger values of h.

Case VI: The benchmark instances considered have to schedule 500 jobs in a single machine for the values of common restrictive due date parameter 'h' taking h = 0.2, 0.4, 0.6 and 0.8, respectively.

We can infer from Table 12 and 13 that % error obtained is high for 500 job size for all values of common due date restrictive parameter 'h'.

Table 7: Job size = 50, h = 0.6 and 0.8

	N = 50, h = 0.6			N = 50, h = 0.8		
Instance	UB	$COST_{GA}$	Error (%)	UB	$COST_{GA}$	Error (%)
k = 1	17,990	18090	0.56	17,990	22183	23.31
k = 2	14,231	14124	-0.75	14,132	17802	25.97
k = 3	16,497	16719	1.35	16,497	23331	41.43
k = 4	14,105	14527	2.99	14,105	20193	43.16
k = 5	14,650	14780	0.89	14,650	21548	47.09
k = 6	14,251	14383	0.93	14,075	18003	27.91
k = 7	17,715	17734	0.11	17,715	23955	35.22
k = 8	21,367	22042	3.16	21,367	30357	42.07
k = 9	14,298	14530	1.62	13,952	16617	19.10
k = 10	14,377	14538	1.12	14,377	19026	32.34

Table 8: Job size = 100, h = 0.2 and 0.4

	N = 100, h = 0.2			N = 100, h = 0.4		
Instance	UB	$COST_{GA}$	Error (%)	UB	$COST_{GA}$	Error (%)
k = 1	156,103	148073	-5.14	89,588	88410	-1.31
k = 2	132,605	126852	-4.34	74,854	75122	0.36
k = 3	137,463	131239	-4.53	85,363	81703	-4.29
k = 4	137,265	131510	-4.19	87,730	81527	-7.07
k = 5	136,761	126061	-7.82	76,424	73196	-4.22
k = 6	151,938	141307	-7.00	86,724	79707	-8.09
k = 7	141,613	137426	-2.96	79,854	79935	0.10
k = 8	168,086	162795	-3.15	95,361	97049	1.77
k = 9	125,153	118870	-5.02	73,605	71695	-2.59
k = 10	124,446	121117	-2.68	72,399	73563	1.61

Table 9: Job size = 100, h = 0.6 and 0.8

	N = 100, h = 0.6			N = 100, h = 0.8		
Instance	UB	$COST_{GA}$	Error (%)	UB	$COST_{GA}$	Error (%)
k = 1	72,019	75867	5.34	72,019	105626	46.66
k = 2	59,351	61542	3.69	59,351	82174	38.45
k = 3	68,537	72085	5.18	68,537	99500	45.18
k = 4	69,231	70863	2.36	69,231	90899	31.30
k = 5	55,291	57379	3.78	55,277	71491	29.33
k = 6	62,519	64064	2.47	62,519	84737	35.54
k = 7	62,213	64118	3.06	62,213	79569	27.90
k = 8	80,844	86108	6.51	80,844	122687	51.76
k = 9	58,771	61139	4.03	58,771	88749	51.01
k = 10	61,419	64503	5.02	61,419	88651	44.34

Table 10: Job size = 200, h = 0.2 and 0.4

	N = 200, h = 0.2			N = 200, h = 0.4		
Instance	UB	$COST_{GA}$	Error (%)	UB	$COST_{GA}$	Error (%)
k = 1	526,666	518411	-1.57	301,449	312439	3.65
k = 2	566,643	561642	-0.88	335,714	336499	0.23
k = 3	529,919	506860	-4.35	308,278	310671	0.78
k = 4	603,709	605475	0.29	360,852	367763	1.92
k = 5	547,953	536046	-2.17	322,268	323575	0.41
k = 6	502,276	497746	-0.90	292,453	296640	1.43
k = 7	479,651	475161	-0.94	279,576	289283	3.47
k = 8	530,896	514133	-3.16	288,746	295619	2.38
k = 9	575,353	548730	-4.63	331,107	324787	-1.91
k = 10	572,866	560805	-2.11	332,808	342565	2.93

Table 11: Job size = 200, h = 0.6 and 0.8

	N = 200, h = 0.6			N = 200, h = 0.8		
Instance	UB	$COST_{GA}$	Error (%)	UB	$COST_{GA}$	Error (%)
k = 1	254,268	274825	8.08	254,268	375591	47.71
k = 2	266,028	286505	7.70	266,028	398791	49.91
k = 3	254,647	274743	7.89	254,647	371561	45.91
k = 4	297,269	314016	5.63	297,269	419893	41.25
k = 5	260,455	284769	9.34	260,455	386855	48.53
k = 6	236,160	259425	9.85	236,160	363254	53.82
k = 7	247,555	268765	8.57	247,555	369427	49.23
k = 8	225,572	245431	8.80	225,572	325683	44.38
k = 9	255,029	275624	8.08	255,029	366687	43.78
k = 10	269,236	289384	7.48	269,236	356555	32.43

Table 12: Job size = 500, h = 0.2 and 0.4

	N = 500, h = 0.2			N = 500, h = 0.4		
Instance	UB	$COST_{GA}$	Error (%)	UB	$COST_{GA}$	Error (%)
k = 1	3,113,088	3305047	6.17	1,839,902	1977577	7.48
k = 2	3,569,058	3699342	3.65	2,064,998	2218344	7.43
k = 3	3,300,744	3429027	3.89	1,909,304	2063522	8.08
k = 4	3,408,867	3545740	4.02	1,930,829	2092647	8.38
k = 5	3,377,547	3433645	1.66	1,881,221	1996460	6.13
k = 6	3,024,082	3091570	2.23	1,658,411	1832905	10.52
k = 7	3,381,166	3502586	3.59	1,971,176	2119823	7.54
k = 8	3,376,678	3490985	3.39	1,924,191	2043356	6.19
k = 9	3,617,807	3704822	2.41	2,065,647	2162955	4.71
k = 10	3,315,019	3437760	3.70	1,928,579	2056772	6.65

Table 13: Job size = 500, h = 0.6 and 0.8

	N = 500, h = 0.6			N = 500, h = 0.8		
Instance	UB	$COST_{GA}$	Error (%)	UB	$COST_{GA}$	Error (%)
k = 1	1,581,233	1843663	16.60	1,581,233	2537261	60.46
k = 2	1,715,332	1967986	14.73	1,715,322	2617714	52.61
k = 3	1,644,947	1938258	17.83	1,644,947	2616006	59.03
k = 4	1,640,942	1921048	17.07	1,640,942	2627574	60.13
k = 5	1,468,325	1675408	14.10	1,468,325	2205653	50.22
k = 6	1,413,345	1678722	18.78	1,413,345	2276736	61.09
k = 7	1,634,912	1953055	19.46	1,634,912	2554413	56.24
k = 8	1,542,090	1805264	17.07	1,542,090	2361021	53.11
k = 9	1,684,055	1964242	16.64	1,684,055	2630581	56.21
k = 10	1,520,515	1762533	15.92	1,520,515	2338440	53.79

Table 14: Job size = 1000, h = 0.2 and 0.4

	N = 1000, h = 0.2			N = 1000, h = 0.4		
Instance	UB	$COST_{GA}$	Error (%)	UB	$COST_{GA}$	Error (%)
k = 1	15,190,371	16053614	5.68	8,570,154	10730700	25.21
k = 2	13,356,727	14250523	6.69	7,592,040	9749841	28.42
k = 3	12,919,259	14170251	9.68	7,313,736	9701558	32.65
k = 4	12,705,290	13753833	8.25	7,300,217	9507948	30.24
k = 5	13,276,868	14595085	9.92	7,738,367	10069800	30.13
k = 6	12,236,080	13533622	10.60	7,144,491	9441817	32.16
k = 7	14,160,773	15072351	6.43	8,426,024	10427617	23.75
k = 8	13,314,723	14041803	5.46	7,508,507	9797581	30.49
k = 9	12,433,821	13900636	11.79	7,299,271	9723774	33.22
k = 10	13,395,234	14278128	6.59	7,617,658	9912807	30.13

Table 15: Job size = 1000, h = 0.6 and 0.8

	N = 1000, h = 0.6			N = 1000, h = 0.8		
Instance	UB	$COST_{GA}$	Error (%)	UB	$COST_{GA}$	Error (%)
k = 1	6,411,581	9774296	52.45	6,411,581	11049760	72.340
k = 2	6,112,598	9248383	51.30	6,112,598	10794675	76.590
k = 3	5,985,538	9443837	57.78	5,985,538	11136987	86.060
k = 4	6,096,729	9210923	51.08	6,096,729	11110292	82.230
k = 5	6,348,242	9425506	48.47	6,348,242	11951954	88.270
k = 6	6,082,142	9115155	49.87	6,082,142	11138921	83.140
k = 7	6,575,879	9677090	47.16	6,575,879	11565517	75.877
k = 8	6,069,658	9096604	49.87	6,069,658	10578145	74.270
k = 9	6,188,416	9274358	49.87	6,188,416	10990477	77.590
k = 10	6,147,295	9580148	55.84	6,147,295	11021781	79.290

Case VII: The benchmark instances considered have to schedule 1000 jobs in a single machine for the values of common restrictive due date parameter 'h' taking h = 0.2, 0.4, 0.6 and 0.8, respectively.

Percentage error obtained is high for 1000 jobs for all values of h can be inferred from Table 14 and 15.

CONCLUSION

In this study, we have carried out several experiments to determine the best crossover rate. Results show clearly that genetic algorithm works well for smaller job sizes 10, 20, 50, 100 and 200, respectively for restrictive common due date parameter h = 0.2 and 0.4 but fails to work for larger job sizes 500 and 1000 for larger values of h. This algorithm can be further extended by incorporating local search techniques to minimize the penalty cost as well as error value.

ACKNOWLEDGMENT

The researcher would like to thank the management of SSN College of Engineering for funding the High Performance Computing Lab (HPC Lab) to carry out this research. The author expresses sincere gratitude to Dr. Chandrabose Aravindan, Professor, Department of CSE, SSN College of Engineering for his valuable guidance throughout the research work.

REFERENCES

Baker, K.R. and G.D. Scudder, 1989. On the assignment of optimal due dates. J. Oper. Res. Soc., 40: 93-95.

Biskup, D. and M. Feldmann, 2001. Benchmarks for scheduling on a single-machine against restrictive and unrestrictive common due dates. Comput. Oper. Res., 28: 787-801.

Cheng, T.C.E. and H.G. Kahlbacher, 1991. A proof for the longest-job-first policy in one-machine scheduling. Nav. Res. Logist., 38: 715-720.

Feldmann, M. and D. Biskup, 2003. Single-machine scheduling for minimizing earliness and tardiness penalties by meta-heuristic approaches. Comput. Ind. Eng., 44: 307-3233.

Hall, N.G., W. Kubiak and S.P. Sethi, 1991. Earliness-tardiness scheduling problems II: Weighted deviation of completion times about a restrictive common due date. Oper. Res., 39(5): 847-856

Hoogeveen, J.A. and S.L. van de Velde, 1991. Scheduling around a small common due date. Eur. J. Oper. Res., 55: 237-242.

James, R.J.W., 1997. Using tabu search to solve the common due date early/tardy machine scheduling problem. Comput. Oper. Res., 24: 199-208.

Le, L. and Z. Hong, 2013. Hybridization of harmony search with variable neighborhood search for restrictive single machine earliness/tardiness problem. Inform. Sciences, 226: 68-92.

Lee, C.Y. and S.J. Kim, 1995. Parallel genetic algorithms for the earliness-tardiness job scheduling problem with general penalty weights. Comput. Ind. Eng., 28: 231-243.

Liao, C.J. and C.C. Cheng, 2007. A variable neighborhood search for minimizing single machine weighted earliness and tardiness with common due date. Comput. Ind. Eng., 52: 404-413.

Lin, S.W., S.Y. Chou and K.C. Ying, 2007. A sequential exchange approach for minimizing earliness-tardiness penalties of single-machine scheduling with a common due date. Eur. J. Oper. Res., 177: 1294-1301.

Nearchou, A.C., 2008. A differential evolution approach for the common due date early/tardy job scheduling problem. Comput. Oper. Res., 35(4): 1329-1343.