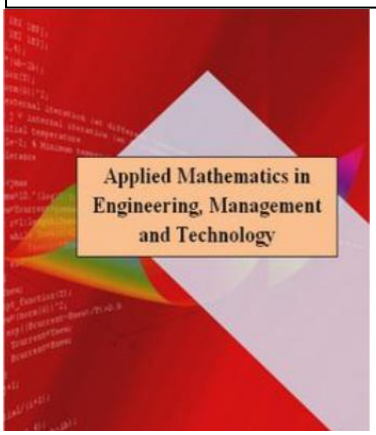
Applied mathematics in engineering, management and technology 5(1) 2017:56-67
www.amiemt-journal.com

A Genetic Algorithm for Minimizing Total Weighted Tardiness on a Single Machine with Sequence Dependent Setup Times

Hossein Zoulfaghari[a1], Javad Nematian[b]
[a] Department of Industrial and Systems Engineering, Isfahan University of Technology, Isfahan, Iran.
Department of industrial engineering, University of Tabriz, Tabriz, Iran.



Abstract

This paper deals with the single machine scheduling problem to minimize the total weighted tardiness in the presence of sequence dependent setup. Setup times are small compared to processing time of jobs, so that they are added to the processing times of jobs. However, in fact these times are exist and should be considered in sequenced jobs. In this paper, firstly, a mathematical model is proposed to describe the problem. According to literature, this problem is a NP-hard, so that a genetic algorithm as a meta-heuristic is proposed. This algorithm is tested on 120 benchmark instances. Results of solving the benchmark instances show that proposed genetic algorithm is an effective and able algorithm for minimizing tardiness on a single machine in the presence of setup times.

Key words: Single Machine, Total Weighted Tardiness, Setup Times, Genetic Algorithm.

1. INTRODUCTION

Many researches in minimizing total weighted tardiness deal with situation where there are no setup times. In this situation the problem is modeled with the assumption that either setup times are small compared to processing time of jobs so that they can be ignored or setup times can be independent of job processing sequence so that they are added to the processing times of jobs. However, substantial setup times are required between the processing of two jobs in some industries and the amount of setup time for a job depends on the preceding job in the processing order. In these cases, setup times should be considered explicitly. Such setup times are called sequence dependent setup times (SDST) and exist in industries that require, for example, cleaning, molding, painting and printing operations (Allahverdi et al., 2008). In the literature for scheduling problems with SDST, different performance measures have been studied, but, minimizing total weighted tardiness (TWT) is very important and in last decades, many researchers has paid attention to it and has reached useful results. This paper investigate with the problem of scheduling jobs with sequence dependent setup times on a single machine to minimize the total weighted tardiness (SMTWT-SDST) , also denoted as $1/s_{ij}/w_jT_j$. This problem has a set of n independent jobs ($N=\{1,2,...,n\}$) which is to be processed without interruption on a single machine. Furthermore, this single machine can process only one job at a time. All jobs can be selected for processing at time zero. Each job $i \in N$ has an integer processing time, a due date, and a positive weight represented by P_i , d_i and w_i ,respectively and a sequence dependent setup time s_{ij} if job j is processed immediately after job i. For a given job sequence, we can complete the earliest completion times C_i and tardiness $T_i=\max\{0,C_i-d_i\}$ for each job $i \in N$. The completion time of job p_i can be computed as $C_k=C_{k-1}+s_{k-1,k}+p_k$. When $k=1$, $C_0=0$ and s_{01} represent the initial setup time of job p_1 . A weighted tardiness penalty w_iT_i will incur when job i is completed after its due date. The objective is to find a job sequence S in order to minimize the sum of weighted tardiness penalty given by the formula $f(S)=\sum_{i=1}^{n} w_iT_i$.

SMTWT problem is strongly NP-hard (Lawler 1977, Lenstra et al., 1977), and thus provides a challenging area for both exact algorithms and meta-heuristic approaches. Lots of researches have been done on this problem and many different approaches have been experimented. One may classify these heuristic algorithms as constructive and improvement heuristics. A sequence is formed by constructive heuristic through assigning a job to the current position at each step of the algorithm according to a dispatching rule. A dispatching rule can

be either static, such as the Earliest Due Date (EDD), or dynamic, such as Apparent Tardiness Cost (ATC) rule. ATC rule was proposed for the single machine total weighted tardiness problem (Vepsalainen & Morton, 1987). Rinnooy Kanetal (1975) extended the dominance rules which are developed by Emmons (1965) for the un-weighted problem to the weighted tardiness problem so that the search could be restricted for an optimal solution. A branch and bound algorithm was combined with Lagrangian relaxation introduced by Potts and Van Wassenhove (1985) which can solve problems with up to 50 jobs to optimality. A survey of dynamic programming and branch and bound algorithms for this problem is given by Abdul-Razaq et al. (1990). They compared these different methods on test problems with up to 50 jobs. The results of comparison showed that these exact algorithms are computationally inefficient when the number of jobs is beyond 50. Therefore, many researchers turn to develop heuristics to obtain near optimal solutions in reasonable time. Cheng et al. (2005) proposed a $O(n^2)$ approximation algorithm and further investigated some special sub-models of this problem to identify polynomially solvable cases. Several construction heuristics and dispatching rules were reviewed by Potts and Van Wassenhove (1991). A priority rule was used by Volgenant and Teerhuis (1999) to improve four construction heuristics. Then they tested them on problems with up to 80 jobs. The simple heuristic methods are fast. But the obvious disadvantage of them is that solutions which are generated by them may be far from the optimum. Many other complex meta-heuristics are also proposed for this problem such as genetic algorithm (Avci et al., 2003), simulated annealing (Potts & Van Wassenhove, 1991), tabu search (TS) (Bozejko et al., 2006, Bilge et al., 2007), ant colony optimization (Holthaus and Rajendran, 2005), and particle swam optimization combining differential evolution algorithms (Tasgetiren et al., 2006). Crauwels et al. (1998) compared the performance of simulated annealing, threshold accepting, TS, and genetic algorithms, using two different solution representations (permutation representation and binary representation). The iterated dynasearch (Congram et al., 2002) is the best algorithm for this problem in the literature, which is a local search technique that uses dynamic programming to find the best move consisting of a series of independent interchange moves and searches an exponential size neighborhood in polynomial time. Later this algorithm is improved by Grosso et al. (2004) through adopting generalized pairwise interchange operators. Though the variable neighborhood search (VNS) has been successfully applied in many combinatorial optimization problems such as the traveling salesman problem, the open vehicle routing problem, the p-median problem, the continuous location allocation problem, and so on (Mladenovic & Hansen,1997 ,Hansen & Mladenovic,2001,Hansen et al., 2008, Fleszar et al.,2007), existing research shows a lack of discussion of VNS on the SMTWT problem. Cicirello and Smith(2005) genera ted 120 problem instances, each with 60 jobs, for the $1/s_{ij}/w_jT_j$ problem and analyzed the effectiveness of stochastic sampling approaches, such as value-biased stochastic sampling (VBSS), a VBSS with hill-climbing, limited discrepancy search, and heuristic-biased stochastic sampling, together with a simulated annealing (SA) approach for the $1/s_{ij}/w_jT_j$ problem. In (Liao & Juan, 2007) and (Anghinolfi & Paolucci, 2008), ant colony optimization (ACO) algorithms were proposed. Lin and Ying compared results of three different meta-heuristics including Genetic Algorithm (GA), Tabu Search (TS) and SA (Lin & Ying, 2007). In (Cicirello, 2006), the results of (Cicirello & Smith, 2006) were improved by means of a GA approach by introducing a non-wrapping order crossover. Later, a beam search with variable beam and filter widths was proposed in (Valente & Alves, 2008). In addition, a discrete particle swarm optimization algorithm (DPSO) was presented with the best known results (Anghinolfi & Paolucci, 2009). Recently, a discrete differential evolution (DDE) algorithm was developed for the $1/s_{ij}/w_jT_j$ problem and the previous best known solutions were improved with excellent results (Tasgetiren et al., 2009). Therefore, in this paper we present a genetic algorithm to solve the $1/s_{ij}/w_jT_j$.

The remainder of the paper is organized as follows. In Section 2, a mathematical programming formulation of the $1/s_{ij}/w_jT_j$ problem is given. Then, the proposed genetic algorithm is explained in Section 3. For more explaining the problem and proposed algorithm, a numerical example is extended. The computational results of the proposed method are presented in Section 5. Finally, conclusions are stated in Section 6.

2. MATHEMATICAL MODEL

The problem $1/s_{ij}/w_jT_j$ can be formulated as a mixed integer linear programming model by defining the following decision variables and considering the notation given earlier:

$$x_{jk}=\begin{cases}1 & If\ \ job\ j\ is\ processed\ at\ the\ k^{th}\ \ position.\\ 0 & Otherwise.\end{cases}$$

$$y_{ijk}=\begin{cases}1 & If\ \ job\ j\ is\ processed\ after\ \ job\ i\ at\ the\ k^{th}\ \ position.\\ 0 & Otherwise.\end{cases}$$

Below we present the mixed integer linear programming model of the $1/s_{ij}/w_jT_j$ problem for the sake of completeness as follows:

$$Min\sum_{j=1}^{n}w_jT_j \qquad (1)$$

s.t :

$$\sum_{k=1}^{n}x_{jk}=1 \qquad \forall j \qquad (2)$$

$$\sum_{j=1}^{n}x_{jk}=1 \qquad \forall k \qquad (3)$$

$$C_j\geq\left(s_{0j}+p_j\right)x_{jk} \quad \forall j\ and\ \ k=1 \qquad (4)$$

$$C_j\geq C_i-M\left(1-y_{ijk}\right)+s_{ij}+p_j \quad \forall i,j,k \quad where\ i\neq j\ and\ \ k\geq 2 \qquad (5)$$

$$T_j\geq C_j-d_j \quad \forall j \qquad (6)$$

$$x_{i(k-1)}+x_{jk}\leq y_{ijk}+1 \quad \forall i,j,k \quad where\ \ i\neq j\ \ and\ k\geq 2 \qquad (7)$$

$$x_{jk}\in\{0,1\} \quad \forall j,k \qquad (8)$$

$$y_{ijk}\in\{0,1\} \quad \forall \mathrm{i},j,k \qquad (9)$$

$$C_j,T_j\geq 0 \quad \forall j \qquad (10)$$

In this model, the total weighted tardiness as the objective function is minimized. Constraint sets (2) ,(3) ensure that each job is assigned to exactly one position and each position in the sequence is assigned to exactly one job, respectively. Constraint set (4) represents the completion time of job j which is assigned to the first position ($k=1$) in the sequence. The completion time of job j (C_j) is calculated in constraint set (5). Constraint set (6) shows the tardiness of the jobs. Constraint set (7) is applied to interrelate two different sets of decision variables. Constraint sets (8) and (9) are integrality constraints, whereas constraint set (10) defines the continuous variables.

As stated in Section 1, the problem is NP-hard and hence solving the above mathematical model to obtain a solution to the problem is not practical. In the following section, we present the genetic algorithm which is developed to find better solutions.

3. GENETIC ALGORITHM

Genetic algorithm as one of the most applicable meta-heuristic methods to solve combinatorial optimization problems was introduced first by Holland (1975). In this section, the suggested genetic algorithm is explained completely. To use genetic algorithm in its best form, first of all, features of this algorithm should be designed based on the problem characteristics. These features are chromosomes (solution presentation), fitness function, initial population, selection of parents for generation of new population, crossover, mutation, and stopping criteria which are explained in the following subsections.

3.1 CHROMOSOME DEFINITION

In proposed GA, chromosomes include n genes which present the number of jobs in definite sequence. It should be mentioned that in chromosome, each gens should satisfy the one job in sequence and amount of gens not be same. Figure 1 represents the chromosomes structure considered for this problem.

2	4	1	5	3

Figure 1. Chromosome (solution representation)

3.2 FITNESS FUNCTION

Fitness function (*ff*) is the value of first objective functions (system availability) plus the penalty of constraints violation. In other words, the problem constraints are added to the objective function in such a way that if one solution goes beyond the constraints, a relatively large amount of penalty is added to the objective function. This penalty keeps the feasibility of the final solution while provides the search in the infeasible space of the problem. The search in the infeasible space leads to an appropriate diversity for genetic algorithm.

3.3 INITIAL POPULATION

In order to produce initial population, *Pop* chromosomes are generated randomly.

3.4 SELECTION

In order to select the required chromosomes in crossover, the following steps are passed. The fitness function (*ff*) is calculated for all the existing chromosomes (*Pop*) in the present population, then from *Pop* present chromosomes, k chromosomes are selected randomly and sorted based on *ff*. The chromosome with largest fitness functions is selected as the parent for generating a new population. This event will happen *Pop* times until finally *Pop* parents are selected for crossover and mutation operators.

3.5 CROSSOVER

Crossover is taken place with a certain rate. By using the crossover, from each two parents, six offspring is generated. The two parents and six offspring create eight chromosomes and the two premier chromosomes based of *ff* are selected to transfer to the next generation. As a result there would be *Pop* population at the end of crossover operations. In order to produce these six offspring from two selected parents, following steps are taken.

Step 1: Two random numbers (m_1, m_2) are selected from 1 to n, where $m_1 < m_2$.

Step 2: For first child, amount of gens from 1 to m_1 be given from first parent and then, amount of other gens are given from second parent, respectively. In this situation, should be careful that amount of gens not be same. For second child, amount of gens from 1 to m_1 be given from second parent and then, amount of other gens are given from first parent, too.

Step 3: For third child, amount of gens from m_1 to m_2 be given from first parent and then, amount of other gens are given from second parent, respectively. In this situation, should be careful that amount of gens not be same. For fourth child, amount of gens from m_1 to m_2 be given from second parent and then, amount of other gens are given from first parent, too.

Step 4: For fifth child, amount of gens from m_2 to n be given from first parent and then, amount of other gens are given from second parent, respectively. In this situation, should be careful that amount of gens not be same. For sixth child, amount of gens from m_2 to n be given from second parent and then, amount of other gens are given from first parent, too.

This kind of crossover leads to increase in capability of the algorithm to find a better solution. In figure 2, an example explains the crossover operation. Suppose that there is a single machine system with 6 jobs. Also, $m_1 = 2, m_2 = 5$:

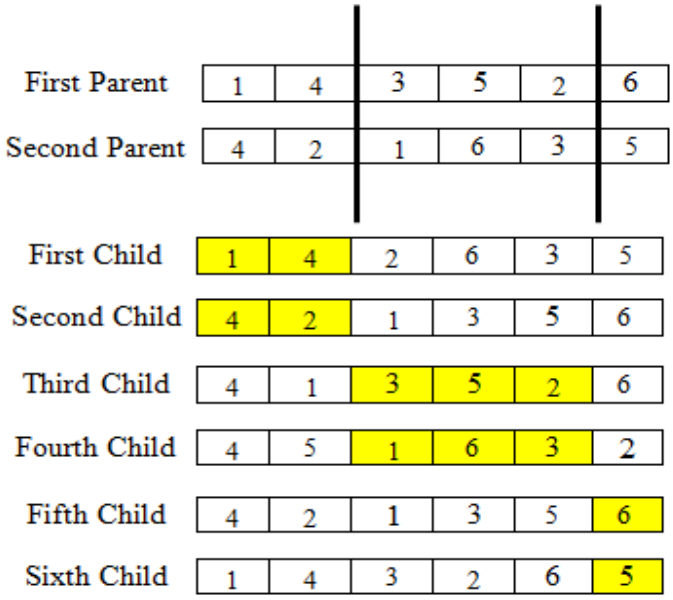


Figure 2. Crossover Operation

Now there are eight chromosomes and we should select two superior ones. For this selection, *ff* is calculated for all these eight chromosomes and they are compared with each other. Finally, two chromosomes with highest *ff* are selected.

3.6 MUTATION

Mutation operator is also used with certain rate which is less than crossover rate. The main purpose of applying mutation operator is to increase diversity and avoid trapping into local optimization. In this operator, one offspring is selected randomly among two chromosomes produced by the crossover operator. Two random numbers $(m_1^{'}, m_2^{'})$ are considered from 1 to n and values of these two genes are exchanged. Then *ff* is calculated for the muted offspring and is compared with the *ff* of the chromosome of pre-mutation. If the *ff* of new offspring is more than the previous one, then it will be replaced by the new generated offspring. Otherwise, the previous offspring remain as the superior ones.

For example, we suppose the fifth offspring has been selected for mutation; $m_1^{'} = 1$ and $m_2^{'} = 4$ .Figure 3 represents the mutation operator for these random values.

Figure 3. Mutation Operation

Crossover and mutation operators and regeneration process transfer one generation to another one.

3.7 STOPPING CRITERIA

The GA process will continue to reach a predefined number of iterations (*Gen*).

4. NUMERICAL EXAMPLE

This part of the paper includes an example which has 5 jobs. The details of example are given in flowing tables.

Table 1. Parameters

		Jobs				
		1	**2**	**3**	**4**	**5**
Parameters	w_i	2	1	1	3	2
	P_i	5	4	4	3	3
	d_i	17	16	20	17	18

Table 2. Setup Times

		Before					
After	**Job**	**0**	**1**	**2**	**3**	**4**	**5**
	1	1	-	1.5	1	2	1
	2	0.5	1	-	2	1	1.25
	3	0.5	1.25	1.5	-	2	0.5
	4	1	1.5	1.25	1.5	-	1
	5	0.5	1	1	0.5	1.5	-

To solve this problem, the proposed genetic algorithm has been used. The optimal scheduling is shown in Figure 4:

1	4	3	5	2	6

Figure 4. Optimal Scheduling

The Total Weighed Tardiness for example equal to 6.25 that proposed algorithm reach to it at a little time. Since size of problem is small, the proposed algorithm can reach optimal solution. But, for problems with medium and large size reach to optimal solution is very hard. To showing efficiency of proposed algorithm, it is tested on 120 benchmark instances.

5. CALCULATING EXPERIMENTS

The performance of the genetic algorithm is compared with that of other heuristic algorithms given in the literature, which were tested on the same benchmark data. The benchmark data include 120 instances, each with 60 jobs and are available at http://www.ozone.ri.cmu.edu/benchmarks.html (Cicirello & Smith, 2006). The instances were generated to cover a wide range of constraints imposed by three parameters, the due date tightness factor τ, the due date range factor R and the setup time severity factor η. The parameters take the following values: $\tau = \{0.3, 0.6, 0.9\}$, $R = \{0.25, 0.75\}$ and $\eta = \{0.25, 0.75\}$. The details of these parameters were explained in (Lee et al., 1997). The following four sets are the best known results so far for the $1/s_{ij}/w_jT_j$ problem:

1. OBK: Overall aggregated best known solutions composed of the solutions generated by the SA, GA, and TS algorithms by Lin and Ying (2007), the ACO algorithm by Liao and Juan (2007) and the GA algorithm by Cicirello (2006).
2. ACO_ AP: Solutions of ACO reported by Anghinolfi and Paolucci (2008).
3. DPSO : Solutions of DPSO presented by Anghinolfi and Paolucci (2009).
4. DDE: Results of DDE given by Tasgetiren et al. (2009).

Results of compare genetic algorithm with above algorithms are shown in Table3.

Table 3. Results of compare genetic algorithm with other algorithms

Instance number	**OBK**	**ACO-AP**	**DPSO**	**DDE**	**Best solution**	**Proposed GA**	$\frac{\textbf{Best solution}-\textbf{GA}}{\textbf{Best solution}} \times 100$
1	684	513	531	474	474	472	0.421940928

2	5082	5082	5088	4902	4902	4894	0.163198694
3	1792	1769	1609	1465	1465	1457	0.546075085
4	6526	6286	6146	5946	5946	5917	0.487722839
5	4662	4263	4339	4084	4084	4281	-4.823702253
6	5788	7027	6832	6652	5788	5734	0.932964755
7	3693	3598	3514	3350	3350	3390	-1.194029851
8	142	129	132	114	114	110	3.50877193
9	6349	6094	6153	5803	5803	5800	0.051697398
10	2021	1931	1895	1799	1799	1804	-0.277932185
11	3867	3853	3649	3294	3294	3271	0.698239223
12	0	0	0	0	0	0	**0 (OP.S*)**
13	5685	4597	4430	4194	4194	4147	1.120648546
14	3045	2901	2749	2268	2268	2381	-4.982363316
15	1458	1245	1250	964	964	951	1.348547718
16	4940	4482	4127	3876	3876	3811	1.676986584
17	204	128	75	61	61	58	4.918032787
18	1610	1237	971	857	857	827	3.500583431
19	208	0	0	0	0	0	**0 (OP.S)**
20	2967	2545	2675	2111	2111	2172	-2.88962577
21	0	0	0	0	0	0	**0 (OP.S)**
22	0	0	0	0	0	0	**0 (OP.S)**
23	0	0	0	0	0	0	**0 (OP.S)**
24	1063	1047	1043	1033	1033	1017	1.548886738
25	0	0	0	0	0	0	**0 (OP.S)**
26	0	0	0	0	0	0	**0 (OP.S)**
27	0	0	0	0	0	0	**0 (OP.S)**
28	0	0	0	0	0	0	**0 (OP.S)**
29	0	0	0	0	0	0	**0 (OP.S)**
30	165	130	0	0	0	0	**0 (OP.S)**
31	0	0	0	0	0	0	**0 (OP.S)**
32	0	0	0	0	0	0	**0 (OP.S)**
33	0	0	0	0	0	0	**0 (OP.S)**
34	0	0	0	0	0	0	**0 (OP.S)**
35	0	0	0	0	0	0	**0 (OP.S)**
36	0	0	0	0	0	0	**0 (OP.S)**
37	755	400	186	107	107	103	3.738317757
38	0	0	0	0	0	0	**0 (OP.S)**
39	0	0	0	0	0	0	**0 (OP.S)**
40	0	0	0	0	0	0	**0 (OP.S)**
41	71186	70253	69102	69242	69102	69008	0.136030795
42	58199	57847	57487	57511	57487	57410	0.133943326
43	147211	146697	145883	145310	145310	144989	0.220907026
44	35648	35331	35331	35289	35289	35194	0.269205701
45	59307	58395	59175	58935	58395	58128	0.457230927
46	35320	35317	34805	34764	34764	34197	1.630997584
47	73984	73787	73378	73005	73005	73004	0.001369769
48	65164	65261	64612	64612	64612	64601	0.017024701
49	79055	78424	77771	77641	77641	77581	0.077278757
50	32797	31826	31810	31565	31565	31419	0.462537621
51	52639	50770	49907	49927	49907	49817	0.180335424
52	99200	95951	94175	94603	94175	94284	-0.11574197
53	91302	87317	86891	84841	84841	84692	0.175622635
54	78960	76503	75490	75367	75367	75362	0.006634203

55	69776	68843	68649	66006	66006	67219	-1.837711723
56	78960	76503	75490	75367	75367	75362	0.006634203
57	67447	66534	64575	64552	64552	64610	-0.089850043
58	48081	47038	45680	45322	45322	45317	0.01103217
59	55396	54037	52001	52207	52001	52197	-0.376915829
60	68851	62828	63342	60765	60765	60769	-0.006582737
61	76369	75916	75916	75916	75916	75881	0.046103588
62	44769	44869	44769	44769	44769	44769	0
63	75317	75317	75317	75317	75317	75317	0
64	92572	92572	92572	92572	92572	92572	0
65	127912	126696	126696	126696	126696	126696	0
66	59832	59685	59685	59685	59685	59751	-0.110580548
67	29390	29390	29390	29390	29390	29390	0
68	22148	22120	22120	22120	22120	22120	0
69	64632	71118	71118	71118	64632	65823	-1.842740438
70	75102	75102	75102	75102	75102	75102	0
71	150709	145825	145771	145007	145007	145007	0
72	46903	45810	43994	43904	43904	43741	0.371264577
73	29408	28909	28785	28785	28785	28785	0
74	33375	32406	30734	30313	30313	30291	0.072576122
75	21863	22728	21602	21602	21602	21602	0
76	55055	55296	53899	53555	53555	53289	0.49668565
77	34732	32742	31937	32237	31937	32108	-0.535429126
78	21493	20520	19660	19462	19462	19462	0
79	121118	117908	114999	114999	114999	114999	0
80	20335	18826	18157	18157	18157	18157	0
81	384996	383485	383703	383405	383405	383405	0
82	410979	409544	409544	409544	409544	409481	0.015382963
83	460978	458879	458787	458752	458752	458752	0
84	330384	329680	329670	329670	329670	329670	0
85	555106	554766	555130	554993	554766	554993	-0.040918153
86	364381	361685	361417	361417	361417	361417	0
87	399439	398670	398551	398670	398551	398558	-0.001756362
88	434948	434410	433519	433186	433186	433186	0
89	410966	410102	410092	410092	410092	410092	0
90	402233	401959	401653	401653	401653	401653	0
91	344988	340030	343029	340508	340030	340500	-0.138223098
92	365129	361407	361152	361152	361152	361152	0
93	410462	408560	406728	404548	404548	403681	0.214313258
94	335550	333047	333298	333020	333020	332981	0.011711008
95	521512	517170	521208	517011	517011	517091	-0.015473559
96	461484	461479	459321	457631	457631	457648	-0.003714783
97	413109	411291	410889	409263	409263	409127	0.033230465
98	532519	526856	522630	523486	522630	523471	-0.160916901
99	370080	368415	365149	364442	364442	364371	0.019481838
100	439944	436933	432714	431736	431736	431736	0
101	353408	352990	352990	352990	352990	352990	0
102	493889	493936	493069	492748	492748	492618	0.026382654
103	379913	378602	378602	378602	378602	378602	0
104	358222	358033	375693	357693	357693	357693	0
105	450808	450806	450806	450806	450806	450806	0
106	455849	455093	455152	454379	454379	454379	0
107	353371	353368	352867	352766	352766	352750	0.004535584

108	462737	461452	460792	460792	460792	460792	0
109	413205	413408	412004	413004	412004	413004	-0.24271609
110	419481	418769	418769	418769	418769	418769	0
111	347233	346763	342752	342752	342752	342752	0
112	373238	373140	369237	367110	367110	367110	0
113	261239	260400	260176	260872	260176	260176	0
114	470327	464734	464136	465503	464136	465492	-0.292155747
115	459194	457782	457874	457289	457289	457108	0.039581096
116	537459	532840	532456	530803	530803	530715	0.016578655
117	512286	506724	503199	502840	502840	502840	0
118	352118	355922	350729	349749	349749	349749	0
119	579462	573910	573046	573046	573046	573058	-0.002094073
120	398590	397520	396183	394183	394183	396183	-0.507378553

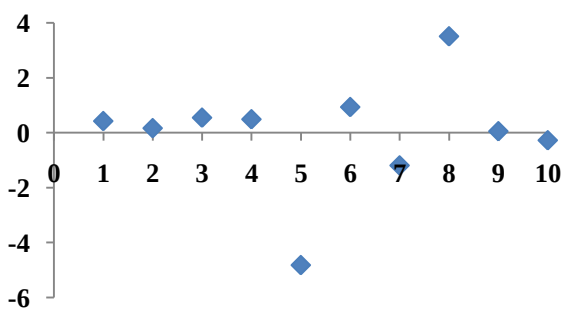
* **OP.S**: Optimal Solution

The results in Table3 show that proposed genetic algorithm is an effective and able approach for minimize the total weighted tardiness in the presence of sequence dependent setup. From Table 3, we can see that the results for 47 (39.17%) out of 120 instances are improved, and for 54 (45%) out of 120 instances are equally solved. The genetic algorithm obtains inferior result only in 19 (15.83%) out of 120 instances compared to all algorithms presented in the literature (that is over all OBK, ACO_AP, DPSO and DDE results). Table4 show the results for coordinating each class.

Table4. Average difference coordinating each class

Class number	Instance number	Parameters values	Average difference
1	1-10	$\tau = 0.3, R = 0.25, \eta = 0.25$	-0.018330
2	11-20	$\tau = 0.3, R = 0.25, \eta = 0.75$	0.539105
3	21-30	$\tau = 0.3, R = 0.75, \eta = 0.25$	0.154889
4	31-40	$\tau = 0.3, R = 0.75, \eta = 0.75$	0.373832
5	41-50	$\tau = 0.6, R = 0.25, \eta = 0.25$	0.340653
6	51-60	$\tau = 0.6, R = 0.25, \eta = 0.75$	-0.204650
7	61-70	$\tau = 0.6, R = 0.75, \eta = 0.25$	-0.190720
8	71-80	$\tau = 0.6, R = 0.75, \eta = 0.75$	0.040510
9	81-90	$\tau = 0.9, R = 0.25, \eta = 0.25$	-0.002730
10	91-100	$\tau = 0.9, R = 0.25, \eta = 0.75$	-0.003960
11	101-110	$\tau = 0.9, R = 0.75, \eta = 0.25$	-0.021180
12	111-120	$\tau = 0.9, R = 0.75, \eta = 0.75$	-0.074550

Figures 5-16 show these results. In these figures, horizontal axis and vertical axis denote the instance number and percent difference, respectively.



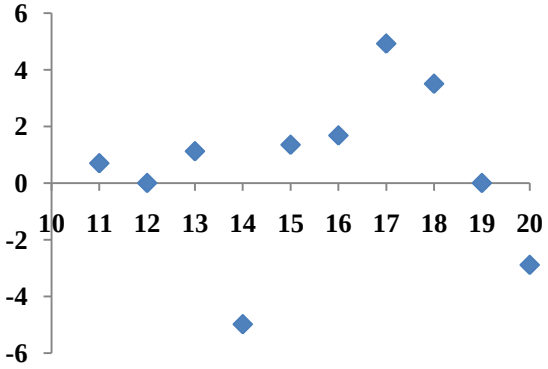


Figure5. Results for Class 1(Instances 1-10)

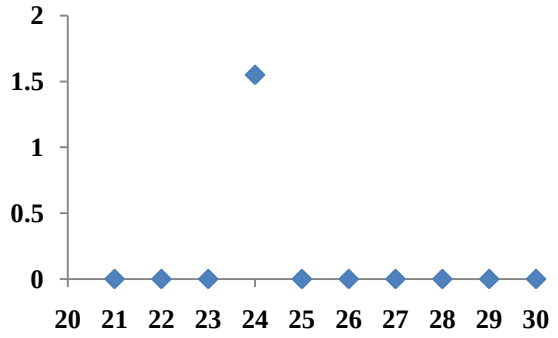


Figure6. Results for Class 2(Instances 11-20)

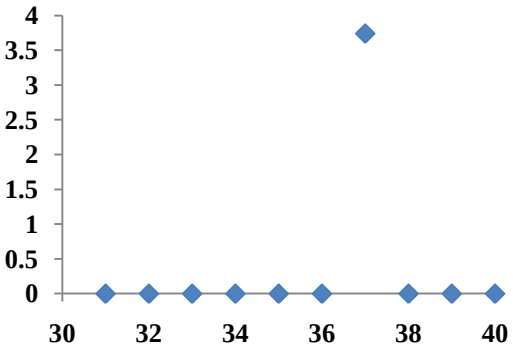


Figure7. Results for Class 3(Instances 21-30)

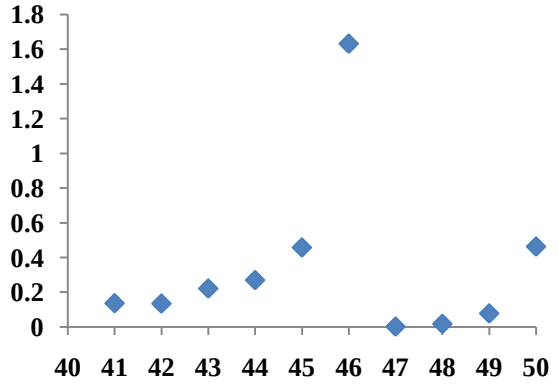


Figure8. Results for Class 4(Instances 31-40)

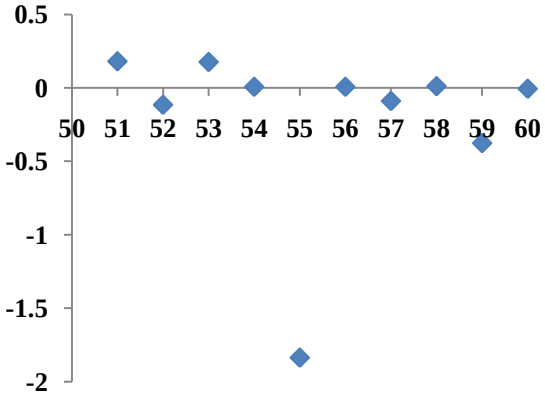


Figure9. Results for Class 5(Instances 41-50)

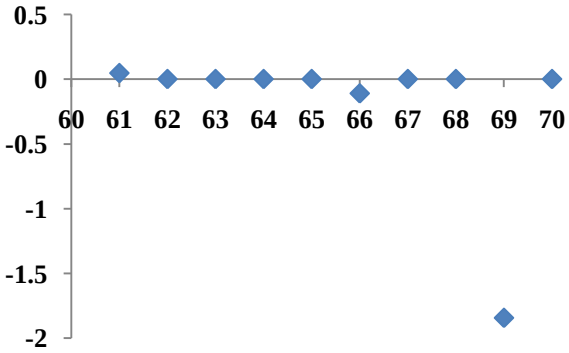


Figure10. Results for Class 6(Instances 51-60)

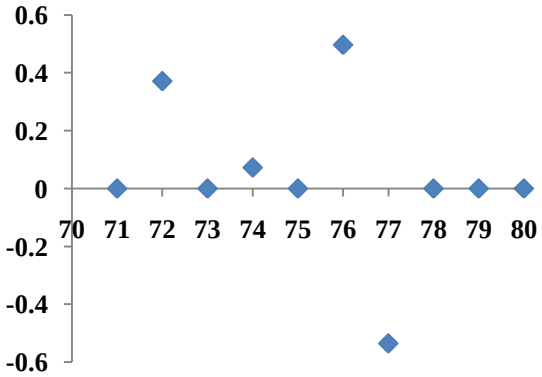


Figure11. Results for Class 7(Instances 61-70)

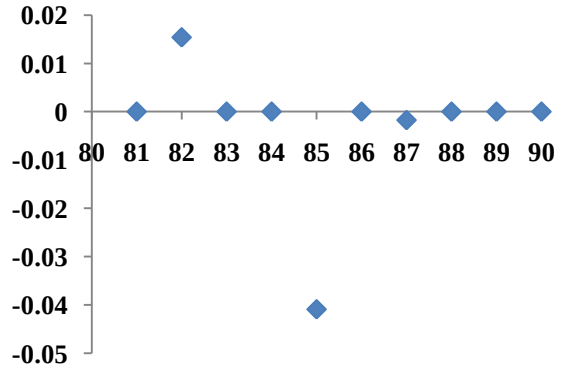


Figure12. Results for Class 8(Instances 71-80)

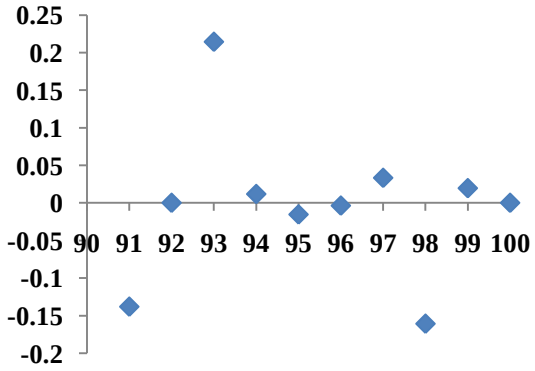


Figure13. Results for Class 9(Instances 81-90)

Figure14. Results for Class 10(Instances 91-100)

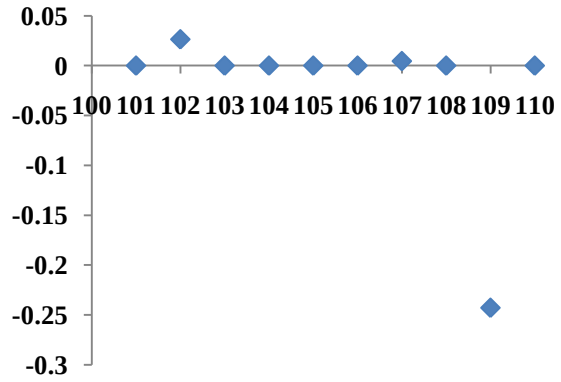


Figure15. Results for Class 11(Instances 100-110)

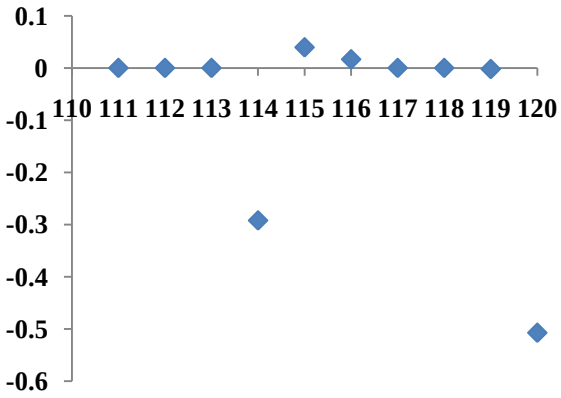


Figure16. Results for Class 12(Instances 111-120)

These figures show that the proposed Genetic Algorithm is effective. Therefore, for classes 1,2,5,8,10 this algorithm improve the best solution.

6. CONCLUSION

In this paper the single machine scheduling problem to minimize the total weighted tardiness in the presence of sequence dependent setup is considered. According to researches, this problem is NP-hard, so that we proposed a genetic algorithm. This algorithm tested on predefined benchmark and compared to four effective algorithms which are existing in literature. This benchmark includes 120 instances, as the results for 47 (39.17%) out of 120 instances are improved and for 54 (45%) out of 120 instances are equally solved. The genetic algorithm obtains inferior result only in 19 (15.83%) out of 120 instances compared to other algorithms. The obtained results show the proposed genetic algorithm is effective and able to find good solution in this field. For future developments and more reality of the modeling and assumptions, we can consider assumption compulsory idle times and setup times depend on sequencing simultaneously.

REFERENCES

Abdul-Razaq, T.S., Potts, C.N., & Van Wassenhove, L.N. (1990). A survey of algorithms for the single machine total weighted tardiness scheduling problem. Discrete Applied Mathematics,26,235–53.

Allahverdi, A., Ng, C.T., Cheng, T.C.E., & Kovalyov, M.Y.(2008). A survey of scheduling problems with setup times or costs. European Journal of Operational Research, 187,985–1032.

Anghinolfi, D., & Paolucci, M.(2008). A new ant colony optimization approach for the single machine total weighted tardiness scheduling problem. International Journal of Operations Research,5(1),44–60.

Anghinolfi, D., & Paolucci, M.(2009). A new discrete particle swarm optimization approach for the single machine total weighted tardiness scheduling problem with sequence dependent setup times. European Journal of Operational Research,193,73–85.

Avci, S., Akturk, M.S., & Storer. R.H.(2003). A problem space algorithm for single machine weighted tardiness problem. IIE Transactions,35,479–86.

Bilge, U., Kurtulan, M., & Kraç, F.(2007). A tabu search algorithm for the single machine total weighted tardiness problem. European Journal of Operational Research,176,1423–35.

Bozejko, W., Grabowski, J., & Wodecki, M.(2006). Block approach tabu search algorithm for single machine total weighted tardiness problem. Computers and Industrial Engineering,50,1–14.

Cheng, T.C.E., Ng, C.T., Yuan, J.J., & Liu, Z.H.(2005). Single machine scheduling to minimize total weighted tardiness. European Journal of Operational Research,165,423–43.

Cicirello, V.A., Smith, S.F.(2005). Enhancing stochastic search performance by value-biased randomization of heuristics. Journal of Heuristics,11,5–34.

Cicirello, V.A. (2006). Non-wrapping order crossover: an order preserving crossover operator that respects absolute position. In: Proceedings of the 8th annual conference on genetic and evolutionary computation, Seattle, Washington, USA, p. 1125–32.

Congram, R.K., Potts, C.N., & Vande Velde, S.L.(2002). An iterated dynasearch algorithm for the single-machine total weighted tardiness scheduling problem. INFORMS Journal on Computing, 14(1),52–67

Crauwels, H.A.J., Potts, C.N., & Van Wassenhove, L.N.(1998). Local search heuristics for the single machine total weighted tardiness scheduling problem. INFORMS Journal on Computing,10(3),341–50.

Emmons, H,.(1969). One machine sequencing to minimize certain functions of job tardiness. Operations Research ,17,701–15.

Fleszar, K., Osman, I.H., & Hindi, K.S.(2008). A variable neighborhood search algorithm for the open vehicle routing problem. European Journal of Operational Research; doi:10.1016/j.ejor.06.0642.
Grosso. A., Della Croce. F., & Tadei. R.(2004). An enhanced dynasearch neighborhood for single-machine total weighted tardiness scheduling problem. Operations Research Letters,32,68–72.
Hansen, P., Mladenovic, N., & Moreno Pérez, J.A.(2008). Variable Neighborhood Search. European Journal of Operational Research,191(3),593–5.
Hansen, P., & Mladenovic, N.(2001). Variable neighborhood search: principles and applications. European Journal of Operational Research,130,449–67.
Holland, J.H.(1975). Adaptation in natural and artificial systems. Michigan: University of Michigan Press.
Holthaus, O., & Rajendran, C.(2005). A fast ant-colony algorithm for single-machine scheduling to minimize the sum of weighted tardiness of jobs. Journal of the Operational Research Society,56,947–53.
Lawler, E.L.(1977). A pseudo polynomial algorithm for sequencing jobs to minimize total tardiness. Annals of Discrete Mathematics,1,331–42.
Lee, Y.H., Bhaskaram, K., & Pinedo, M.(1997). A heuristic to minimize the total weighted tardiness with sequence-dependent setups. IIE Transactions,29,45–52.
Lenstra, J.K., Rinnooy Kan, A.H.G., & Brucker, P.(1977). Complexity of machine scheduling problems. Annals of Discrete Mathematics,1,343–62.
Liao, C.J., Juan, H.C.(2007). An ant colony optimization for single machine tardiness scheduling with sequence dependent setups. Computers & Operations Research,34,1899–909.
Lin, S.W., & Ying, K.C.(2007). Solving single-machine total weighted tardiness problems with sequence-dependent setup times by meta-heuristics. International Journal of Advanced Manufacturing Technology,34,1183–90.
Mladenovic, N., & Hansen, P.(1997). Variable neighborhood search. Computers & Operations Research,24,1097–100.
Potts, C.N., & Van Wassenhove, L.N.(1985). A branch and bound algorithm for the total weighted tardiness problem. Operations Research,33,363–77.
Potts, C.N., & Van Wassenhove, L.N.(1991). Single machine tardiness sequencing heuristics. IIE Transactions,23,346–54.
Rinnooy Kan, A.H.G., Lageweg, B.J., & Lenstra, J.K.(1975). Minimizing total costs in one-machine scheduling. Operations Research,25,908–27.
Tasgetiren, M.F., Liang, Y.C., Sevkli, M., & Gencyilmaz, G.(2006). Particle swarm optimization and differential evolution for the single machine total weighted tardiness problem. International Journal of Production Research,44,4737–54.
Tasgetiren, M..F, Pan, Q.K., & Liang, Y.C.(2009). A discrete differential evolution algorithm for the single machine total weighted tardiness problem with sequence dependent set up times. Computers & Operations Research,36(6),1900–15.
Valente, J.M.S., & Alves, R.A.F.S.(2008). Beam search algorithms for the single machine total weighted tardiness scheduling problem with sequence-dependent setups. Computers & Operations Research,35(7),2388–405.
Vepsalainen, A.P.J., & Morton, T.E.(1987). Priority rules for job shops with weighted tardiness cost. Management Science,33(8),1035–47.
Volgenant, A., & Teerhuis, E.(1999). Improved heuristics for the n-job single machine weighted tardiness problem. Computers & Operations Research,26(1),35–44.