Gazi University Journal of Science
GU J Sci
25(1):127-136 (2012)

ORIGINAL ARTICLE

A New Crossover Operator for Single Machine Total Weighted Tardiness Problem with Sequence Dependent Setup Times

Gökhan KIRLIK[1,♠], Zuhal KARTAL[2], Servet HASGUL[3]

[1]*College of Engineering, Koç University, Rumelifeneri Yolu, 34450, Sariyer, Istanbul, Turkey.*

[2]*Department of Industrial Engineering, Anadolu University, 26555, Eskisehir Turkey.*

[3]*Department of Industrial Engineering, Eskişehir Osmangazi University, 26480, Eskisehir Turkey.*

Received: 11.03.2011 Revised: 14.03.2011 Accepted: 31.03.2011

Abstract

The single machine total weighted tardiness problem with sequence dependent setup times is a challenging and heavily studied problem. This problem is NP-hard, so several heuristics have been proposed in the literature so far. One of them is the genetic algorithm. The genetic algorithm is both powerful solution technique and applicable to wide range of different problem types, although its performance is heavily parameter and operator dependent. It is seen in literature that the well-conducted and adapted genetic algorithm operators and parameters increase the solution quality. In this study, a new crossover operator is proposed for the single machine with sequence dependent setup times problem to minimize the total weighted tardiness. The proposed crossover operator improves the relative positions by using apparent tardiness cost with setups (ATCS) heuristic while preserving the absolute positions. These are the two main aspects of the permutation type crossover operators for scheduling problems. The performance of the proposed crossover operator is tested by comparing it with partially mapped crossover (PMX) in different test cases using benchmark instances from literature. It is shown that the proposed ATCS based crossover operator gives better results than PMX in all test problems.

Key Words: *Weighted tardiness scheduling, Sequence dependent setups, Genetic algorithm, Crossover operator*

1. INTRODUCTION

There has been a vast body of research on scheduling problems for more than fifty years. Among them, single-machine scheduling is one of the most popular problems. A group of machines may possibly be treated as a single unit in the context of single-machine scheduling problem. For scheduling purposes, computer-controlled machining centers and robotic cells are often evaluated as single-machine scheduling problems.

Panwalkar et al. reveals the fact that, about three quarters of managers scheduled at least one operation that required sequence-dependent setup times; whereas only 15% of them stated that all operations required sequence-dependent setup times [1]. There are two types of scheduling problems that involve setup times: the ones that are sequence-dependent and the sequence-independent ones. The setups are sequence dependent when the setup time of a job depends on the job that is processed immediately before it on the machine. If the setup time depends only on the job that the machine is being setup for, but not on the previously processed job, then that the setups are sequence-independent. There are many examples of sequence-dependent setups in different fields of the industry like stamping operation in plastic manufacturing, roll slitting in the paper

♠Corresponding author, e-mail: gkirlik@ku.edu.tr

industry and die changing in a metal processing shop. In [2] a review of scheduling research that includes the setup time considerations is presented.

As the just-in-time (JIT) production philosophy spread out over the last two decades, many researchers worked on scheduling problems to meet the due date requirements. The underlying principle of JIT is producing goods only when it is necessary. Panwalkar et al. defined meeting due dates as the most important scheduling criterion for JIT problems [1].

A due-date related performance measure is the minimization of total weighted tardiness. In manufacturing systems, this is achieved by using specific penalty (weight) functions. This paper considers single-machine total weighted tardiness problems with sequence-dependent setup times (STWTSDS). This problem is represented as $1|s_{ij}|\sum w_j T_j$ by using three-field notation. The special case of this problem is denoted as $1|s_{ij}|\sum T_j$ and it is proved as NP-hard [3, 4]. Therefore, STWTSDS problem is also NP-hard.

In the literature, both exact and heuristic algorithms are proposed for the single machine total weighted tardiness problem [5-7]. However, exact algorithms can only solve small sized instances, since the required computational effort is huge. Consequently, the design of heuristics has become the main aspect in recent studies. These heuristics can be classified as constructive and improvement heuristics. Constructive heuristics use dispatching rules to determine a complete job sequence. At each step, a new job is fixed in a position and this position cannot be changed at later steps. The dispatching rule can be a static one such as earliest due date (EDD) or a dynamic one such as the apparent tardiness cost (ATC) rule. ATC is proposed for the single machine total weighted tardiness (STWT) problem in [8]. Up to now, the best-known construction heuristic for single machine total weighted tardiness with sequence-dependent setup time problem given in the literature is the apparent tardiness cost with setups (ATCS) rule [9]. Although, ATCS is very fast to develop a feasible solution, the solution quality is quite unsatisfactory, especially for large-sized problems. The other type of heuristics is the improvement heuristics, which start with an initial solution and may provide better solutions by rearranging the sequence. In [10], several heuristics are compared for single machine total tardiness problem. It is shown that the pairwise interchange methods outperform the other methods. In [9], two types of improvement methods proposed for single machine total weighted tardiness with sequence-dependent setup time problem which are swap and insertion based hill-climbing search procedures.

In addition to the constructive and the improvement heuristics, metaheuristics are extensively studied recently in the literature to solve this problem. A metaheuristic is a framework of the general algorithm and it can be used in different optimization problems with small changes. In [11], 120 problem instances were generated for STWTSDS problem, each with 60 jobs. The effectiveness of stochastic sampling approaches as value-biased stochastic sampling (VBSS), a VBSS with hill-climbing (VBSS-HC), limited discrepancy search (LDS), heuristic-biased stochastic sampling (HBSS) and a simulated annealing (SA) approach [11] are discussed for this problem. In [12] and [13] ant colony optimization (ACO) algorithms are proposed. Lin and Ying compared results of three different metaheuristics including Genetic Algorithm (GA), Tabu Search (TS) and SA [14]. In [15], Cicirello's best known results are improved by means of a GA approach by introducing a non-wrapping order crossover (XO). A beam search with variable beam and certain filter widths is proposed in [16]. In addition, a discrete particle swarm optimization algorithm (DPSO) is presented with the best known results [17]. Recently, a discrete differential evolution (DDE) algorithm is developed for this problem and the previous best known solutions were improved with excellent results [18].

In STWTSDS literature, GA is one of the common solution techniques used to solve the problem. In fact, GA is the most widespread tool used to generate solutions for combinatorial optimization problems. Although, GA is a powerful solution technique, its performance is operator-dependent, that is, the results are subject to the nature of selection, crossover and mutation operators. Especially, crossover operator affects the quality of the solution deeply [19]. In this study, a problem specific crossover operator called as ATCS-based crossover is proposed. This operator is developed by combining good aspects of the partially mapped crossover operator (PMX) and the ATCS rule. Meanwhile the absolute positions of the genes inherited from parents are preserved by PMX, ATCS rule improves the solution. Performance of the proposed and the PMX operators are compared by using of the benchmark problems from [11]. 64,800 different tests are realized to reveal the effectiveness of the proposed crossover operator combining of it by three different mutation operators.

This paper will proceed as follows. In section 2, mathematical model of STWTSDS is given. In section 3, genetic algorithms are explained briefly. In section 4, PMX and proposed ATCS-based crossover operators are given. This is followed by computational results in section 5. Finally, conclusions are presented.

2. PROBLEM FORMULATION

In the mathematical formulation, we want to obtain a sequence of jobs, so that binary variable $x_{ij} = 1$ if job *j* is scheduled immediately after job *i*, otherwise $x_{ij} = 0$ where $i, j \in J$. If the first scheduled job is job *j*; the variable $x_{0j} = 1$, otherwise $x_{0j} = 0$. If the last scheduled job is job *j*;

the variable $x_{j,n+1}=1$, otherwise $x_{j,n+1}=0$. Additionally, $s_{0j}=0$ is the setup time for job *j*, if the job *j* scheduled in first position in the sequence. Mathematical model of the STWTSDS problem is given below.

Sets

J: Jobs set $\{1,2,...,n\}$

Indexes

i,j The job index used as a unique identifier for each job, $i, j \in J$

Parameters

n : The number of jobs

p_j : Processing time of the job *j*, $j \in J$

w_j : Weight (priority) of the job *j*, $j \in J$

d_j: Due date of the job *j*, $j \in J$

s_{ij} : Sequence dependent setup time of job *j* if job *i* precedes job *j* $i, j \in J$

Decision Variables

$$x_{ij}=\begin{cases}1 & \text{If job } j \text{ processed after job } i\\ 0 & \text{otherwise}\end{cases}$$

C_j : The completion time of the job *j*, $j \in J$

T_j : Tardiness of the job *j*, $j \in J$

Model

$$\min\left(\sum_j w_j \cdot T_j\right) \quad (1)$$

Subject to,

$$\sum_{\substack{i\cup\{0\}\\ i\neq j}} x_{ij}=1\ ,\ \forall j\in J \quad (2)$$

$$\sum_{j\in J} x_{0j}=1 \quad (3)$$

$$\sum_{j\in J} x_{j,n+1}=1 \quad (4)$$

$$\sum_{\substack{i\cup\{0\}\\ i\neq j}} x_{ij}-\sum_{\substack{i\cup\{n+1\}\\ i\neq j}} x_{ji}=0\ ,\forall j\in J \quad (5)$$

$$C_j \geq \sum_{i\cup\{0\}} (C_i + s_{ij} + p_j)\, x_{ij}\ ,\forall j\in J \quad (6)$$

$$T_j \geq C_j - d_j,\ \ \forall j\in N \quad (7)$$

$$x_{ij}\in\{0,1\},\qquad \forall i\in J\cup\{0\},\ \forall j\in J\cup\{n+1\} \quad (8)$$

The objective function (1) minimizes total weighted tardiness of the jobs. Constraint (2) ensures that each job is processed exactly once. Constraint (3) guarantees that exactly one job is assigned to first position. Similarly, constraint (4) ensures that exactly one job is assigned to last position. Constraint set (5) is flow conservation constraint that ensures the jobs are correctly ordered within a machine schedule. Equation (6) defines the completion time of the jobs. Equation (7) defines the tardiness of the jobs. Constraint (8) is the integrality for the decision variables. This is a non-linear integer model, which is not easy to solve.

3. GENETIC ALGORITHMS

Genetic algorithms are defined as general purpose population based search techniques and their logic depends on the principle of natural selection and natural genetics [20]. The concept of GA is introduced by Holland [21]. It has been used in many optimization problems so far [20, 22]. Michalewicz, also showed good examples to employ GAs to problems encountered in production systems and operations research areas [23].

The idea of GA is based on the populations of potential solutions, which are called chromosomes. These set of potential solutions are recombined or evolved by using of genetic operators. These genetic operators are selection, crossover and mutation. Each individual chromosome has its own fitness value. The selection operator randomly chooses amongst the best-fitted chromosomes based on a fitness criterion and executes the reproduction process. Then, new individuals are created by crossover and mutation operators following reproduction process.

Crossover operators are used to exchange information between parent chromosomes. This operator takes the genetic information of two parent chromosomes and recombines them to produce children chromosomes for the next generation to get better properties than of those parents. Mutation operator is executed selectively after crossover, so the solution is prevented from focusing into a narrow area of the search space or getting stuck into a local optimum.

4. PMX AND ATCS-BASED CROSSOVER

4.1. Partially Mapped Crossover

Goldberg and Lingle proposed the partially mapped crossover for permutation coding [24]. In this operator, two crossover points are selected randomly and the substrings defined by these two points are called mapping sections. The mapping relationship between these two sections is determined. One of these mapping sections is directly inherited to the child chromosome from one parent by considering predetermined mapping relationship; the remaining elements are inherited from the other parent.

In partially mapped crossover, two offsprings are copied from parents P_1 and P_2, called O_1 and O_2, and *i*-th index value of P_1 and P_2 are represented as $v_1(i)$ and $v_2(i)$, respectively. Randomly, two positions, *a* and *b*, are selected uniformly from interval (1,*L*), where *L* is the length of the permutations. Let's assume ($a \leq b$). The values of the *a*th positions of parents P_1 and P_2 are $v_1(a)$ and $v_2(a)$, respectively. These values are determined in O_1 and O_2, and they are swapped. This swapping

operation is repeated for all $v_1(a+i)$ and $v_2(a+i)$ in O_1 and O_2 where $i \in \{0,1,\ldots,(b-a)\}$. The remaining elements, which become the duplicates of the swapped elements in O_1, are changed according to the predetermined swapping relationship. An example for the PMX operator is given in Fig. 1. Here, the parent chromosomes, which are generated randomly (Fig1.a), consist of 15 genes (number of jobs to be scheduled). The segment is chosen between 6th and 11th positions. Sequentially, segments are swapped without breaking mapping rule given in Fig. 1.c. Obtained offspring chromosomes (by using of PMX operator) O_1 and O_2 from example parents P_1 and P_2 are given in Fig. 1.b.

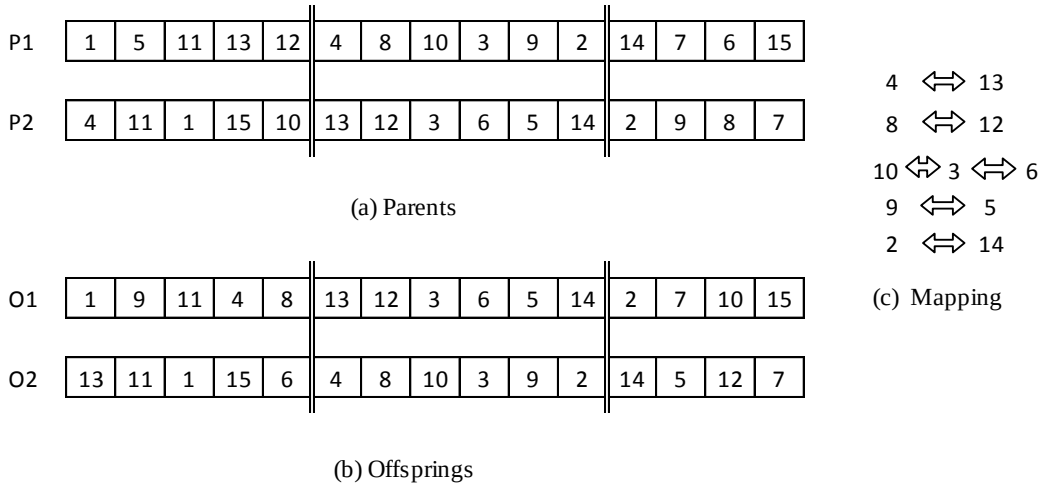

Figure 1. Example for PMX

4.2. ATCS-based Crossover

For some problems represented as permutations, the key building blocks of a solution are derived from the absolute position in the permutation. For others, the relative positions are the most important [15]. For STWTSDS problem solutions are affected by both the absolute and the relative positions at the same time. Absolute positions are important because of the fact that the jobs with very high weights and early due dates need to occur towards the beginning of the sequence. Additionally, because of the sequence-dependent nature of the setup times, the relative positions of the jobs are also important.

In GA literature, some crossover operators preserve absolute positions while, some of them preserve relative positions. Cycle crossover is one of the position based crossover operators that preserves the absolute positions of the jobs without considering their relative positions. On the other hand, the order crossover intends to preserve relative order of jobs. However, to the best knowledge of authors, in literature there is no crossover operator that aims both to preserve absolute position and relative positions simultaneously.

In this study, a new crossover operator is proposed that increases the performance of the solution by preserving both absolute and relative positions. For this purpose, ATCS heuristic is used, proposed by Lee et al. in 1997. Essentially, ATCS heuristic generates a schedule by using job order priority considering weights, sequence dependent setups and due dates. Below, the working logic of ATSC is explained.

ATCS consists of two stages. The first stage of ATCS includes running statistical analysis of the problem instances that are generated randomly. In this stage, formulas are determined to estimate three factors that define the problem instance and makespan. It is most likely a pre-processing stage. In the second stage, by use of these estimations, the look-ahead parameter values are calculated. These values are used for a given problem to apply the ATCS priority index. The importance of these priority indices are biased on the fact that they are used to determine the sequence of the jobs when assigned to the machines that are available at the earliest time point. The following three factors quantify the problem characteristics:

a. Due date tightness
b. Due date range
c. Setup time severity factor

The first factor is due date tightness τ, which is given in equation (9).

$$\tau = 1 - \frac{\bar{d}}{C_{\max}} \qquad (9)$$

In this equation, *Cmax* is the makespan, which is the completion time of the last job to leave the system, and $\bar{d}$ is the average of the due dates. Due to the sequence dependent setup times, determining the makespan value beforehand is difficult. Estimation of the $C_{\max}$ value is explained below. τ gives a information about the tightness of the due dates. The second factor is due date range (R), which is described in equation (10).

$$R = \frac{d_{\max} - d_{\min}}{C_{\max}} \quad (10)$$

In this equation, $d_{\max}$ shows the maximum due date value, similarly $d_{\min}$ shows the minimum due date value. *R* provides an empirical measurement about spread of the due dates. As due date tightness factor, $C_{\max}$ estimation is necessary to determine due date range factor. The next factor is setup time severity factor, (η), which is defined in equation (11).

$$\eta = \frac{\bar{s}}{\bar{p}} \quad (11)$$

In this equation, $\bar{s}$ represents average setup time and $\bar{p}$ represents average processing time. η describes the relative importance of the setup times compared to the processing times.

Finally, it is necessary to determine makespan value of the instance to calculate τ and *R* coefficients. Lee et al. proposed an equation to estimate the completion time of the last processed job that is given in equation (12). Obtained makespan value is correlated with average processing time, average setup time and β. Variability of setup times and number of jobs in the instance affect the value of β (1997).

$$C_{\max} = n(\bar{p} + \beta\bar{s}) \quad (12)$$

By using of these factors, the heuristic determines the parameters k_1 and k_2, which affect the performance of the priority rule. The equations of k_1 and k_2 are given in (13).

$$\begin{aligned} k_1 &= 4.5 + R \quad R \le 0.5 \\ k_1 &= 6.0 - 2R \quad R > 0.5 \\ k_2 &= \frac{\tau}{2\sqrt{\eta}} \end{aligned} \quad (13)$$

First step of the ATCS is completed with determination of k_1 and k_2 parameters. In the second step, the priority index employed by ATCS heuristic is calculated as follows:

$$I_j(t,l) = \frac{w_i}{p_i} \exp\left[-\frac{\max(d_j - p_j - t, 0)}{k_1 \bar{p}}\right] \exp\left[-\frac{s_{lj}}{k_2 \bar{s}}\right] \quad (14)$$

In equation (14), *t* denotes the current time and *l* the index of the job just completed; $\bar{s}$ is the average setup time; k_1 and k_2 are scaling parameters. The ATCS rule separates the effect of the setup time. The priority of a job given by weighted shortest processing time ratio is exponentially discounted twice, once based on slack and again based on setup. These two effects are scaled separately by the parameters k_1 and k_2, which jointly provide the look-ahead capabilities of the ATCS rule. The values of the parameters depend on the problem instance as they essentially perform the scaling.

The proposed ATCS-based crossover operator uses a modified version of ATCS heuristic and PMX. This operator follows the logic of PMX, it retains the absolute positions as well as improving the positions replaced (both absolute and relative) by use of ATCS heuristics.

In ATCS-based heuristic crossover, two children are copied from parents P1 and P2, called O1 and O2, respectively. Randomly, two positions, a and b, are selected uniformly from interval *(1,L)*, where *L* is the length of the permutations. Let's assume $a \le b$. Offspring O1 is searched for *v2(a), v2(a + 1), . . . , v2(b)* between *[1,(a-1)]* and *[(b+1), L]*. Those locations are replaced by holes. From all jobs, scheduled jobs in O1 are removed to determine unscheduled job list for the O1. This procedure is repeated by inserting holes in O2 in the place of values *v1(a), v1(a+1), . . . , v1(b)*. The values *v1(a),v1(a+1),..,v1(b)* are directly inherited to the offspring O2 by preserving their actual order and values. Then, these holes in offspring O1 are filled with unscheduled jobs by using of the modified ATCS rule. The same procedure is applied for offspring O2. The ATCS heuristic generates whole schedule by calculating job scheduling order priorities. On the other hand, the modified ATCS determines the best fit job for each hole position from unscheduled job list. Briefly, for each hole position in the offspring chromosomes, priorities are determined by using of equation (14) for all unscheduled jobs. Then, the highest priority-unscheduled job is assigned to that hole position and removed from unscheduled jobs list. When the unscheduled job list is empty, the procedure is ended.

An example is given in Figure 2 for the proposed crossover operator. The problem parameters (process time, weight, sequence dependent setup, due date) are necessary to apply ATCS-based crossover. Hence, first 15 job parameters of the first instance are used from 120 STWTSDS problem benchmark instances. In the example parameters, due dates were divided by 5. Firstly, two offspring (O1 and O2) are copied from parents P1 and P2. Then, in O1 between positions [1, 5] and [12, 15] search for segment of P2 which is {13, 12, 3, 6, 5, 14}. Those locations (2, 4, 5, 12, 14) are replaced by holes and unscheduled jobs list is obtained {4, 8, 9, 10}. After that, values between [6, 11] in P2 are assigned to O1 between [6, 11] (Figure 2.b). Finally, hole locations are filled by using modified ATCS heuristic with unscheduled jobs. Consequently, offsprings are obtained by ATCS heuristic are given in Fig. 2.c.

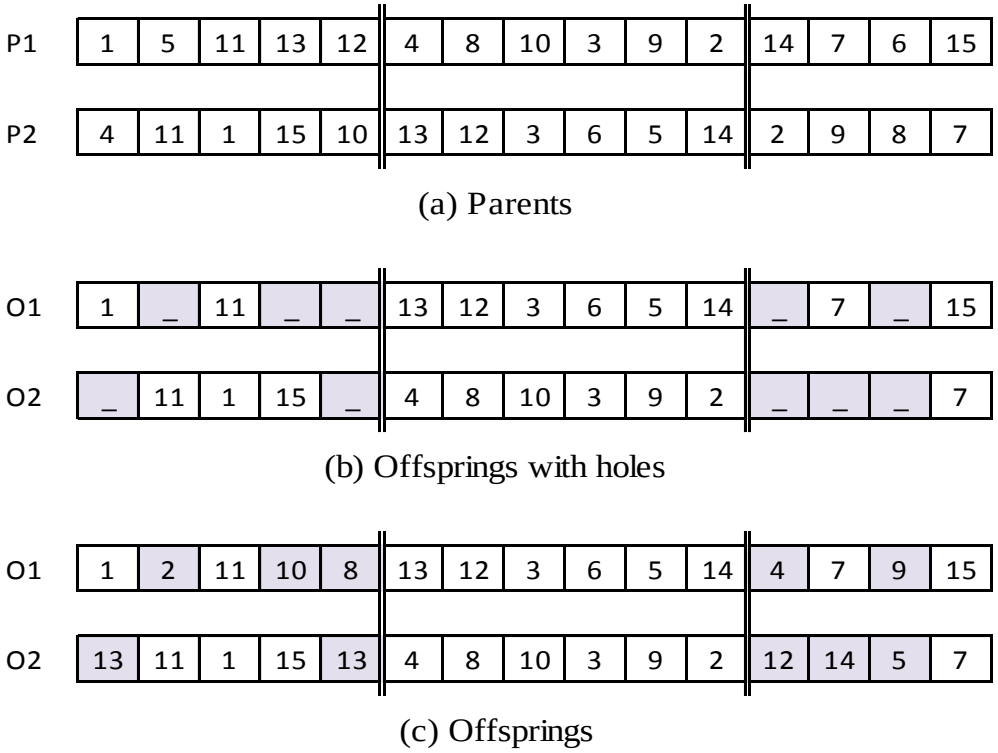

Figure 2. Example for the ATCS-based Crossover

5. COMPUTATIONAL RESULTS

The proposed crossover operator, PMX and other operators of the genetic algorithm are coded in C++ and run with an Intel Pentium Core2 Duo 2.0 GHz at 2048 MB RAM under the Linux OS. Performance comparison of ATCS-based crossover and PMX were realized by using of 120 benchmark instances, each with 60 jobs [11]. Three parameters characterize each problem instance: the due-date tightness factor (τ), the due-date range factor (R) and the setup time severity factor (η). The benchmark set is established by the following parameter values: τ ={0.3,0.6,0.9}, R ={0.25,0.75},and η ={0.25,0.75}. Each of the 12 combinations of parameters leads to 10 problem instances. The benchmark problems can be obtained from OR-lib.

5.1. Improving Performance of ATCS heuristic

The results obtained by ATCS are parameter dependent, so an analysis is conducted to evaluate the improvement of these parameters. In [9], three different parameters are defined for deriving k_1 and k_2 parameters. These three parameters are due date tightness factor, due date range factor and setup times' severity factor. Due date tightness and due date range factors are dependent to $C_{\max}$. $C_{\max}$ is calculated by equation (12) . As seen from the equation β coefficient affects the result of the makespan. Lee et al. proposes coefficient of variation of setup times and number of jobs to determine β [9]. Since β is highly correlated with setup times, it is better to test different β values to improve the results. For this purpose, different β values are tested starting from 0 to 1 by increasing 0.01 at each step. Additionally, k_1 and k_2 parameters, which directly affect the solution of ATCS heuristic are tested to improve the solution of ATCS heuristic. k_1 is tested as starting from 1 to 6 by increasing by 0.25 at each step and k_2 is tested starting from 0.1 to 0.9 by increasing them by 0.05 at each step.

In this study, objective function values are compared according to the ATCS sequence. Best results obtained from both β search and k_1, k_2 parameter search are compared with proposed method in [9]. The results are reported in Table 1.

Table 1. Improvements in ATCS heuristic

Instance Group	Instance Number Range	τ	R	η	Beta Search	k_1 and k_2 Search
1	1-10	0.3	0.25	0.25	23.780	65.658
2	11-20	0.3	0.25	0.75	32.287	79.180
3	21-30	0.3	0.75	0.25	52.503	146.015
4	31-40	0.3	0.75	0.75	25.148	93.851
5	41-50	0.6	0.25	0.25	6.384	17.469
6	51-60	0.6	0.25	0.75	6.284	29.653
7	61-70	0.6	0.75	0.25	3.418	20.219
8	71-80	0.6	0.75	0.75	12.435	35.527
9	81-90	0.9	0.25	0.25	0.053	3.795
10	91-100	0.9	0.25	0.75	0.758	6.370
11	101-110	0.9	0.75	0.25	0.015	1.555
12	111-120	0.9	0.75	0.75	1.631	9.929

5.2. Comparing Performance of Crossover Operators

In this subsection, comparative test results between ATCS-based XO and PMX are presented. For the comparison of crossover operators of GA approach, the chromosome representation, reproduction strategy, crossover operators, mutation operators and termination condition are described as follows:

Chromosome representation: In this study, permutation coding is used to represent a feasible schedule. The chromosome consists of an order of jobs where the *j*th number in the permutation denotes that the job is the *j*th job to be processed. Each chromosome in the initial population is generated randomly. Population size is 30.

Reproduction Strategy: The objective function of the STWTSDS problem is minimization of total weighted tardiness value. So that, the smaller the total objective function value means higher fitness value of the chromosome. During the evaluation process, a super chromosome may be obtained that dominates the population. To avoid this circumstance and to transform minimization type objective function into the fitness maximization type, a scaling operator (normalizing) is used as [25].

$$f_k' = \frac{f_{\max} - f_k + \gamma}{f_{\max} - f_{\min} + \gamma} \quad (15)$$

In the equation (15), f_k is the total weighted tardiness value of the chromosome, $f_{\max}$ and $f_{\min}$ represent maximum and minimum total weighted tardiness value of the population respectively, γ is a very small positive number, and f_k' is the scaled fitness value of the chromosome.

Brindle's selection operator is used to obtain the next generation in the GA [26]. This selection operator works along deterministically. It determines the number of copies of each chromosome in the population by using expected value of the individuals and then copies selected chromosomes for the next generation.

Crossover Operators: During the tests to show the effectiveness of the ATCS-based XO, the result of the proposed crossover is compared with PMX. Therefore, two different crossover operators used are ATCS-based XO and PMX. Besides, different crossover rates are tested to present effectiveness of the proposed crossover in different crossover ratios. 0.8, 0.9 and 1.0 are used as crossover rates.

Mutation: The proposed ATCS-based XO and PMX operator is tested with different mutation operators with different mutation rates. Three different mutation operators are used are *insertion mutation*, *swap mutation* and *swap-insertion mutation*. As insertion mutation is concerned, a random gene and a position are selected, and the selected gene is removed from original place and inserted into a randomly selected position. As for swap mutation, two genes are selected and the content of the selected genes are swapped. Finally, swap-insertion mutation applied either insertion mutation or swap mutation with 50% probability. In addition, these mutation operators are tested with different mutation rates, which are 0, 0.005 and 0.01. The meaning of the 0 mutation rate is mutation operator is not applied. Summary of the results are given in Fig. 3.

Termination Condition: GA is terminated after certain iterations. In all tests, GA is terminated after 500 generations.

		Mutation Rates		
		0	*0.005*	*0.010*
Crossover Rates	*0.8*	55.22	58.33	57.08
	0.9	65.13	59.48	60.161
	1.0	61.66	64.08	60.29

% difference

		Mutation Rates		
		0	*0.005*	*0.010*
Crossover Rates	*0.8*	-0.42	-0.45	-0.45
	0.9	-0.48	-0.51	-0.52
	1.0	-0.52	-0.56	-0.57

Execution time difference

Swap Mutation

		Mutation Rates		
		0	*0.005*	*0.010*
Crossover Rates	*0.8*	54.92	95.13	100.14
	0.9	47.29	116.15	149.05
	1.0	69.25	204.5	153.52

% difference

		Mutation Rates		
		0	*0.005*	*0.010*
Crossover Rates	*0.8*	-0.42	-0.43	-0.46
	0.9	-0.47	-0.49	-0.52
	1.0	-0.52	-0.54	-0.59

Execution time difference

Insertion Mutation

		Mutation Rates		
		0	*0.005*	*0.010*
Crossover Rates	*0.8*	47.69	53.57	87.87
	0.9	56.37	81.02	100.06
	1.0	60.62	101.76	112.53

% difference

		Mutation Rates		
		0	*0.005*	*0.010*
Crossover Rates	*0.8*	-0.42	-0.44	-0.46
	0.9	-0.47	-0.5	-0.52
	1.0	-0.52	-0.53	-0.57

Execution time difference

Swap & Insertion Mutation

Figure 3. Mutation operators test results

$$D = \frac{Z_{PMX} - Z_{ATCS}}{Z_{ATCS}}$$

$$t_d = t_{PMX} - t_{ATCS} \qquad (16)$$

In Table 2, results of the ATCS-based XO and PMX operators with different mutation rates are given. In this table, average relative differences and execution times are presented. Relative differences and execution time differences are calculated with equation (16). In this equation, *D* represents the relative difference between PMX and ATCS-based crossover, *Z* represents the objective value with the given crossover operator, and *t* represents the execution time of the given crossover operator. Both operators are tested on 120 benchmark instances with 27 different cases (three different mutation rates, three different crossover rates and three different mutation operators). Besides, each test is replicated 10 times. Therefore, during the tests 64,800 runs are applied to compare the results of the crossover operators.

As seen from the table, superiority of ATCS-based XO against PMX is clear. Especially, differences are peak when the insertion mutation is applied. On the other hand, computation time of the ATCS-based XO is a little bit longer compared to PMX execution time.

6. CONCLUSION

In this study, a new crossover operator is proposed for STWTSDS problem. Thanks to this proposed operator, formation of illegal chromosomes is avoided while absolute positions of parent chromosomes are preserved. Here, ATCS heuristic that determines job order according to priority of jobs for STWTSDS problem by considering the problem parameters is used to change the absolute and relative position of genes to improve the solution by using of ATCS priority rule.

The performance of ATCS heuristic is changed depending on job priority that in turn depends on k_1 and k_2 parameters and β. Therefore, a search to determine optimal parameters is done first to improve the performance of ATCS heuristic. Firstly, a beta value search is realized which coefficient is used to estimate the makespan. The second type of search is conducted to find the k_1 and k_2 parameters to minimize the obtained objective function. As a result of the search procedures, considerable improvements are obtained (an average of 13.73% for the first type and an average of 42.44% for the second type searches). In addition to, an amount of improvement up to 146.02% is observed in a sample group.

Finally, to show the effectiveness of the proposed ATCS based crossover operator, this crossover operator was compared by PMX by using three different mutation operators with three different mutation rates (0, 0.005, and 0.01). These mutation operators are swap and an insertion based search and a heuristic mutation that combine swap and insertion. In this phase, 64,800 tests are realized. In all test results, the proposed ATCS based crossover gave better results in terms of solution quality though its solution time is slightly higher than that of PMX. In addition, the differences between the performance of two crossover operators is observed as 204.5% when the mutation rate is 0.005 by using of insert mutation and crossover with a rate of 1.0.

For further studies, ATCS based crossover operator can be compared with other permutation type crossover operators which can be used for STWTSDS problem.

REFERENCES

[1] Panwalkar S. S., Dudek R. A., Smith M. L., "Sequencing research and the industrial scheduling problem" ***Symposium on the Theory of Scheduling and its Applications***, p. 29B—38, New York, USA, Springer (1973).

[2] Allahverdi A., Ng C. T., Cheng T., Kovalyov M., "A survey of scheduling problems with setup times or costs" ***European Journal of Operational Research***, 187:985–1032, (2008).

[3] Eugene L. Lawler., "A pseudopolynomial algorithm for sequencing jobs to minimize total tardiness" ***Annals of Discrete Mathematics***, 1:331–342, (1977).

[4] Lenstra J. K., Hendrik A., Kan G. R., Brucker P., "Complexity of machine scheduling problems", ***Annals of Discrete Mathematics***, 1:343–362, (1977).

[5] Abdul-Razaq T. S., Potts C., Van Wassenhove L., "A survey of algorithms for the single machine total weighted tardiness scheduling problems", ***Discrete Applied Mathematics***, 26:235–253, (1990).

[6] Pott C. N., Van Wassenhove L., "A branch and bound algorithm for the total weighted tardiness problem", ***Operations Research***, 26:363–377, (1985).

[7] Pott C. N., Van Wassenhove L., "Dynamic programming and decomposition approaches for the single machine total tardiness problem", ***European Journal of Operational Research***, 32:405–414, (1987).

[8] Vepsalainen A., Morton T. E., "Priority rules for job shops with weighted tardiness cost", ***Management Science***, 33(8): 1035–1047, (1987).

[9] Hoon L. Y., Bhaskaram K., Pinedo M., "A heuristic to minimize the total weighted tardiness with sequence-dependent setups", ***IIE Transactions***, 29:45–52, (1997).

[10] Potts C. N. Van Wassenhove L., "Single machine tardiness sequencing heuristics", ***IIE Transactions***, 23(4):346–354, (1991).

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

[12] Liao C., Juan H., "An ant colony optimization for single machine tardiness scheduling with sequence dependent setups", ***Computers & Operations Research***, 34:1899–1909, (2007).

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

[14] Lin S., Ying K., "Solving single-machine total weighted tardiness problems with sequence-dependent setup times by meta-heuristics", ***International Journal of Advanced Manufacturing Technology***, 34:1183–1190, (2007).

[15] Cicirello V., "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***, p. 1125–1132, Seattle, Washington, USA, (2006).

[16] Valente J., Rui A. F. S., "Beam search algorithms for the single machine total weighted tardiness scheduling problem with sequence-dependent setups", ***Computers & Operations Research,*** 35(7):2388–2405, (2008).

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

[18] Tasgetiren M., Pan Q., Liang Y., "A discrete differential evolution algorithm for the single

machine total weighted tardiness problem with sequence dependent setup times", ***Computers & Operations Research***, 36(6):1900–1915, (2009).

[19] Gen M., Cheng R., "Genetic Algorithms and Engineering Optimization", ***John Wiley Sons Inc.***, NY, USA, (2000).

[20] Goldberg D. E.,. ***Genetic Algorithms in Search, Optimization and Machine Learning***. Addison-Wesley, Reading, MA, USA, (1989).

[21] Holland J. H., "Adaptation in Natural and Artificial Systems"**,** ***University of Michigan Press***, Ann Arbor, MI, USA, (1975).

[22] Davis L., "Applying adaptive algorithms to epistatic domains", In ***Proceedings of the 9th international joint conference on Artificial intelligence***, p. 162–164, Los Angeles, California, USA,. Morgan Kaufmann Publishers Inc, (1985).

[23] Michalewicz Z., "Genetic Algorithms + Data Structures = Evolution Programs", ***Springer***, Berlin, 2nd edition edition, (1994).

[24] Goldberg D. E., Lingle R., "Alleles, loci, and the traveling salesman problem", ***Proceedings of the First International Conference on Genetic Algorithms and Their Applications***, p. 154–159, Mahwah, NJ, USA, Lawrence Erlbaum Associates, Publishers 1985.

[25] Cheng R., Gen M., "Evolution program for resource constrained project scheduling problem", In ***Proceedings of the 1st Conference on Evolutionary Computation***, p. 736–741, Mahwah, NJ, USA, (1994).

[26] Brindle A., "Genetic Algorithms for Function Optimization", PhD thesis, ***University of Alberta***, Edmonton, Canada, (1981).