International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 22 (2017) pp. 12538-12545

A Heuristic Approach for Single Machine with Different Due Dates and Setup Times to Minimize Total Earliness and Tardiness

Sukjae Jeong

College of Business Administration, Kwangwoon University
20 Kwangwoon-ro, Nowon-gu, Seoul, South Korea.
(Orcid: 0000-0001-7081-7567)

Abstract

In order to minimize total earliness and tardiness with different due dates and setup times, the problem of scheduling a given number of jobs on a single is frequently discussed in the industrial field. In this paper, we proposes an efficient genetic algorithm designed to address scheduling issues by minimizing the sum of early and tardy production in a single machine while considering at once the arrival time and delivery. In order to validate the effectiveness of the proposed genetic algorithm, a small-scale problem is compared against the optimal solution of the enumeration method, while a large-scale problem is compared against the solution of the Tabu search, an intelligent search technique.

Keywords: Single machine scheduling, Earliness and Tardiness, Different due dates, Setup time. Genetic algorithm

INTRODUCTION

The manufacturing environment of today has witnessed ever-diversifying customer demand for products, irregular orders, and complicated changes in manufacturing conditions. In order to address this changing environment, manufacturing is evolving from the mass production of yesterday to an individual production or flexible manufacturing pattern. The purpose of the production management section in this manufacturing pattern is to establish an efficient production schedule that adequately reflects the actual circumstances of manufacturing so that delivery can be made precisely on time as demanded by customers. Therefore, corporations are keen on enhancing management of delivery to customers. The most ideal form of production planning is to ensure that all jobs are delivered precisely on time as specified by customers. However, due to various constraints, it is realistically difficult to satisfy customers' delivery needs for all jobs involved. In fact, most make-to-order manufacturing firms are facing this crucial issue, and they typically rely on empirical methods to address it.

Despite that most make-to-order manufacturers are confronted by the aforementioned issue, a systematic approach to establish efficient production scheduling is yet to be developed and prior research in this area has been insufficient. Previous studies on scheduling only dealt with single performance measures, which are often referred to as regular measures, and most of the literature covers scheduling issues with the purpose of minimizing average handle time, average delay time, total handling time, and the number of instances of delayed delivery. Therefore, most evaluations are made based on whether the delivery is made on time. This concept cannot be overlooked under the JIT manufacturing environment. Fry et al. (1990) attributes this to the fact that both early production and tardy production incur cost. Therefore, for ideal scheduling, all jobs should be delivered accurately and on time as specified by the customer. Issues like this can be expressed as problems that have various objective functions whose purpose is to minimize deviation between due dates and completion time. However, in order to devise realistic scheduling, it is desirable to present a realizable solution that considers irregular delivery, arrival time, etc. simultaneously.

This study proposes a genetic algorithm designed to address scheduling issues by minimizing the sum of early and tardy production in a single machine while considering at once the arrival time and delivery. In order to validate the effectiveness of the proposed genetic algorithm, a small-scale problem is compared against the optimal solution of the enumeration method, while a large-scale problem is compared against the solution of the Tabu search, an intelligent search technique.

The paper is organized as follows: Chapter 2 provides a literature review, Chapter 3 explains the hypotheses for problem definition and creates an arithmetic model, Chapter 4 describes the proposed genetic algorithm, Chapter 5 analyzes the performance of the proposed genetic algorithm through a computer experiment, and Chapter 6 provides a summary.

LITERATURE REVIEW

A large number of papers have been published on scheduling models with a single machine scheduling problem. Sadykov (2008) proposed the hybrid branch-and-bound algorithm for solving the scheduling problem of minimizing the weighted number of late jobs on a single machine. Valente and Goncalves (2009) considered a single machine scheduling problem with linear earliness and quadratic tardiness cost and no machine idle time. Jeong and Kim (2008) established the parallel machines scheduling problem in which n jobs having

different release times, due dates, and space limits. In order to solve the problem, a heuristic is developed which is divided into three modules: job selection, machine selection, and job sequencing. Low and Hsu *et al.* (2010) presented a partial swarm optimization (PSO) algorithm for solving the single machine scheduling problem with periodic maintenance activities. Kim and Yano (1994) discussed characteristics of the optimal solution for a single machine problem when the arrival time for individual jobs is the same while the due dates are different, and suggested a heuristic algorithm to address this issue. Soroush (2007) studied a static stochastic single machine scheduling problem in which jobs have random processing times with arbitrary distribution, due dates are know with certainty, and fixed individual penalties are imposed on both early and tardy jobs. Mazzini and Armentano (2001) developed a constructive heuristic algorithm, which determine the sequence of jobs and simultaneously insert idle time for solving the problem of scheduling jobs with distinct ready times and due dates in a single machine in order to minimize the total earliness and tardiness penalties. Valante and Alves (2005) considered the single machine earliness/tardiness scheduling problem with different release dates and no unforced idle time. They decomposed this problem into weighted earliness and weighted tardiness sub problems, and presented lower bounding procedures for each of these sub problems and two dominance rules derived for the problem with equal release dates in order to eliminate dominated nodes from the search tree.

PROBLEM DEFINITION

This study considers an actual scheduling problem for producing high-priced products with strong make-to-order characteristics. Because the product characteristics are high-cost and technology-intensive, early production leads to issues of inventory cost and utilization of storage space, whereas delayed production affects the overall ship building schedule and leads to payment of compensation to customers for the delay. Therefore, the important issue in this plant would be to determine how precisely each job involved could meet their respective due dates. In order for each job to meet the deadline, determination of the time to start assembly at the plant becomes the most crucial matter. Therefore, the reasonable time to start assembly is determined by establishing scheduling for the test-board within the assembly plant (Jeong and Kim, 2008).

In this study, customer orders are assumed to be the jobs. In addition, these jobs each have different arrival times and due dates. If each job fails to meet the delivery on time, it will incur cost for both early production and tardy production, leading to high cost. Therefore, the purpose is to minimize the deviation between the due date and the work completion time.

In this study, this problem is defined as a MINSUM-SM-ET model. Garey et al. (1988) demonstrated for the first time that establishing a schedule for different jobs in a single machine is an NP-hard problem, when the objective function minimizes the sum of early and tardy production, and the jobs have different due dates but the same arrival time. Therefore, the scheduling issue described in this study, in which each job has different due dates but the same arrival time and the sum of early and tardy production is minimized, can be easily defined as NP-hard, as it is a generalization of the arrival time of the issue discussed by Garey et al. (1998).

The following hypotheses are developed to define the model introduced in this study.

(1) Each job can be launched at any point of time after the arrival time.

(2) An on-going job cannot be stopped for the sake of another even when an urgent job
arises.

(3) The cost for early and tardy production for all jobs is assumed to have the same rate.

$$Minimize\ f(s) = \sum_j \{\alpha E_j + \beta T_j\} \qquad (1)$$

Subject to

$$c_j \geq r_j + p_j, \qquad j \in J \qquad (2)$$

$$c_j + E_j - T_j = d_j, \qquad j \in J \qquad (3)$$

$$c_k - c_j + \Phi(1 - y_{jk}) \geq p_k, \qquad j = 1,...,k,...,n-1, k = j+1,...,k,...,n \qquad (4)$$

$$c_j - c_k + \Phi(y_{jk}) \geq p_j, \qquad j = 1,...,k,...,n-1, k = j+1,...,k,...,n \qquad (5)$$

$$E_j, T_j, c_j \geq 0, \qquad j \in J \qquad (6)$$

Equations (1) through (6) show the aforementioned models expressed as mixed integer programming. Equation (1) shows an objective function that minimizes the sum of early and tardy production. This is expressed as a function for the completion time, because the value diminishes as the completion time of each job approaches the due date. Therefore, a more desirable approach is to make the completion of each job as close to the due date as possible. In Equation (1), $\sum_j (\alpha E_j + \beta T_j)$ means $\sum_j x_j (\alpha E_j + \beta T_j)$.

Equation (2) indicates that the arrival time of a job is considered when determining the completion time of that job. This also means that the job cannot be launched before it arrives. Equation (3) shows that the early production time or

the tardy processing time is calculated based on the completion time. Equations (4) and (5) establish the relationship of completion time between job j and job k when job j precedes job k. Here the inequality sign denotes that idle time is allowed when scheduling. If the order of job j and job k switches, Equations (4) and (5) become redundant. Equation (6) shows non-negativity conditions.

The arithmetic models above are NP-hard issues inclusive of integer-type variables. As it is very difficult to obtain a solution when the size of the problem is large, a method to efficiently obtain the solution is developed in the next section. Furthermore, this study assumes that the unit cost of early and tardy production of all jobs involved is the same ($a = b$) in order to suggest the characteristics of the solution and the solution method.

DESIGN OF GENETIC ALGORITHM

Gen and Cheng (1997) noted that a genetic algorithm can provide a strong tool to obtain an optimal solution or a similar optimal solution of a combinatorial optimization issue in many areas, including scheduling, group technology, sales people, etc. This method performs a search in a way that is similar to a biological evolution process in order to improve upon a solution after starting from a randomly selected, initial group of solutions called a population. Each element comprising the population is called a chromosome, and the elements comprising each chromosome are called genes.

The most important aspect in solving various problems with a genetic algorithm is to represent the solution to the given problem naturally and to design genetic operators to ensure similar optimal solutions can be obtained. More specifically, representation, initialization, evaluation function, crossover, mutation, and selection of the solution comprise the process. In addition, genetic parameters including population size (pop_size), maximum number of generations ($\max\_gen$), crossover probability (p_c) and mutation probability (p_m), and they must be defined prior to implementation of the genetic algorithm.

Solution representation

This study uses permutation encoding, which is represented by a job list. Random keys are used to obtain the initial population to produce various entities. This is used to express a solution by generating random numbers between (0,1) and by organizing these values. For instance, the chromosomes for six jobs are comprised as follows:

$$Random\ key\ list: [0.15, 0.89, 0.13, 0.97, 0.56, 0.78]$$

Here, location j within the list denotes job j, and the random number in position j denotes the order of job j. The following job list is obtained when the random key list is reorganized in ascending order:

$$Job\ list: [3, 1, 5, 6, 2, 4]$$

The jobs are represented with integers. The initialization process generates random numbers for the size of the population to comprise initial possible solutions.

Fitness Function

Chromosomes of the population are evaluated by using an evaluation criterion called fitness in order to improve on the solution from generation to generation. Because the objective of most optimization issues is to maximize utility, the evaluation criterion of the fitness level is calculated based on the original objective function. However, because the issue covered in this study is to minimize the objective function, it must be transformed to an issue of maximizing the form of the objective function in order to calculate the evaluation criterion of the fitness level for each chromosome. The objective function is transformed as shown below as the evaluation criterion is used as a fitness function in this paper.

$$fit(c_k) = 1/\ f(c_k), \qquad k = 1, 2, ..., pop\_size \tag{7}$$

$fit(c_k)$ of Equation (7) represents the fitness function of the k^{th} chromosome, while $f(c_k)$ denotes the value of the objective function for the k^{th} chromosome. In order to obtain the objective function, the start and completion time of each job must be determined as follows: the start and completion timings are determined in a reverse manner by aligning the completion time to the due date of each job in the order of the current list while the order of the genes (jobs) within each randomly generated job list (chromosome) is acknowledged as it is. Equation (8) provides a numerical representation of this method. Here, $[i]$ means job assigned by i^{th} ($i = 1, 2, ..., n$).

$$c_{[i]} = \begin{cases} c_{[i-1]} + p_{[i]}, & if\ d_{[i]} - c_{[i-1]} < p_{[i]} \\ d_{[i]}, & otherwise \end{cases} \tag{8}$$

Genetic Operator

Crossover and mutation operators are genetic operators used to comprise a group of more suitable and desirable solutions for the next generation from the group of possible solutions for the current generation. The crossover operator creates new solutions by exchanging a number of genes of the parent chromosomes with one another. The mutation operator, on the other hand, plays a role of helping improve solutions by changing the chromosome structure very slightly. In this paper, a new crossover operator and mutation operator are developed. They fundamentally generate chromosomes for two offspring from the parent chromosomes. The procedures of developing the new crossover and mutation operators are as follows:

Crossover procedure

Step1. Choose at random two points from the parents. If the locations of the two points selected are the same, choose another point at random instead so that the locations are not the same.

Step2. The locations of jobs between the two randomly chosen points are inherited unchanged to the first offspring chromosome.

Step3. The remaining genes (jobs) that are not inherited from the first parents are inherited to offspring chromosomes by moving them from left to right from the second parents to generate the first offspring chromosomes.

Step4. The second offspring chromosome is based on the second parents instead of the first parents in Step 1, and the second parents in Step 3 are changed into the first parents, and the remaining process takes place in the same manner.

Step5. Generations are created repeatedly ($pop\_size \times p_c$) times. Here, pop_size represents the number of chromosomes generated for one generation, or the population size.

Figures 1 and 2 illustrate how two sets of offspring chromosomes are generated from two respective sets of parent chromosomes. Figure 1 is the process for generating the first offspring chromosomes while Figure 2 illustrates the procedure to generate the second offspring chromosomes.

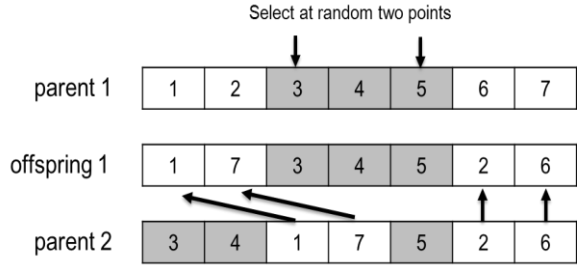

Figure 1: Example of crossover operator to generate the first offspring chromosomes

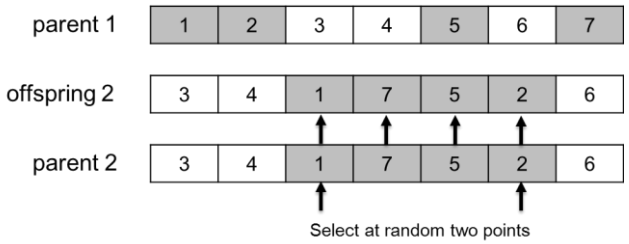

Figure 2: Example of crossover operator to generate the second offspring chromosomes

Mutation procedure

Step 1. Select at random two points in the same chromosome. The gene at the first point is called the chosen gene and the second point is where the chosen gene should be inserted. Here, if the point of the chosen gene and the point for insertion are the same, the point for insertion shall be chosen randomly again.

Step 2. If comparison of the point of the chosen gene and the point for insertion finds reveals that the point index for the chosen gene is greater than the point index for insertion, then the chosen gene is inserted into the point for insertion and the remaining genes then move from left to right. Otherwise, the chosen gene is inserted into the point for insertion and then the remaining genes move from right to left.

Step 3. Generations are created repeatedly for ($pop\_size \times p_m$) times.

Figures 3.3 and 3.4 illustrate the procedure for the mutation operator. The notable feature of this operator is that it has bi-directional characteristics, meaning that the direction of movement is determined by the point of the chosen gene and the point of the gene to be inserted. Figure 3.3 shows the chromosome generation process when the point index for the chosen gene is the 6th, and the point index for insertion is the 3rd. Figure 3.4 shows the chromosome generation process when the point index for the chosen gene is the 3rd, and the point index for insertion is the 6th.

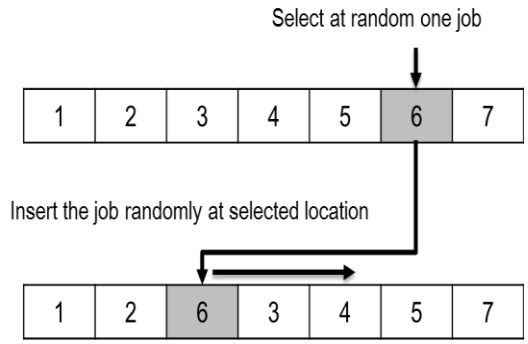

Figure 3: Mutation procedure (a)

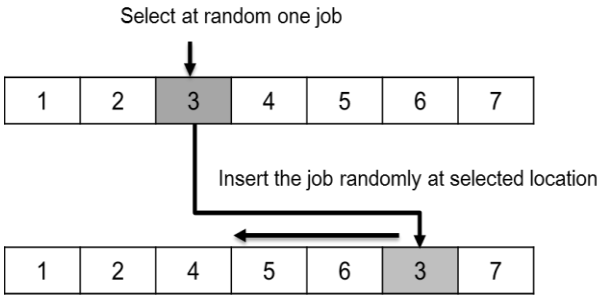

Figure 4: Mutation procedure (b)

Selection Criteria

The selection criteria define how to select chromosomes from the current parent population in order to generate the next-generation offspring. They are chosen based on the fitness of their parent chromosomes. The roulette wheel method introduced in this paper entails prejudice in securing diversity of solutions, as it selects solutions with relatively high suitability. Therefore, this study uses a combination of *remainder stochastic sampling without replacement* (RSSWR) (Goldberg 1989) and the elitism method, which ensures that the chromosome that generated the best solution in the previous generation is always inherited to the next generation.

A brief description of the procedure involving RSSWR is as follows: The fitness of each chromosome is measured and a normalized value for each fitness level is obtained. These normalized values are then represented with an integer unit and a prime number unit, and chromosomes are inherited to the offspring generation as many times as the integer. The same procedure is repeated as many times as the size of the population starting from the chromosomes with greater fitness and proceeding to those with lower fitness. If generation equating to the size of the population cannot be done with the integer unit only, the prime number units of the chromosomes arranged in the order of fitness are used for inheritance to the offspring generation.

Heuristic Procedure

The overall procedure of the genetic algorithm suggested in this paper is summarized as follows:

Step1. Initialization

The initial solution is generated at random as many times as the pop _ size.

Step2. Evaluation

The objective function of Equation (1) is calculated for each work list (chromosome) of the current population, and its fitness is evaluated by using Equation (7).

Step3. Computation of normalized values

To correct statistical errors when choosing job lists from the current generation to the next, normalized values (η_k) are computed for each job list by Equation (9).

$$\eta_k = \{ fit(c_k) / \sum_{k=1}^{pop\_size} fit(c_k) \} \times pop\_size \quad (9)$$

Step 4. Reproduction

The population is regenerated as many times as *pop_size* at the current generation, in order to inherit superior work lists to the next generation by using the RSSWR method.

Step 5. Recombination

Operations are done as many times as crossover probabilities and mutation probabilities, in order to generate the population for the next generation from the reproduced population in Step 4. Also, the best work list of the current generation is inherited to the next generation to create a new population.

Step 6. Termination

The procedure stops when the pre-set number and time of iteration are satisfied. Otherwise, it is iterated from Step 2.

COMPUTATIONAL EXPERIMENTS

To evaluate the performance of the proposed genetic algorithm, this section considers the situations of H Company's assembly plant for ship engines. The actual time required to assemble a ship engine is about 30 to 40 days on average, though it may vary slightly depending on the engine type. The arrival of jobs takes place throughout the year, and the average allowed slack time is 20 days until a job arrives at the assembly plant before it is completed, exclusive of the assembly time. Given these situations, the conditions for the experiment are set as follows:

(1) The assembly time (p_j) for jobs is generated at random from a discrete uniform distribution between [30, 40].

(2) The arrival time (r_j) of jobs is generated at random from a discrete uniform distribution between [1, 365].

(3) The delivery time (d_j) of jobs is generated at random from a discrete uniform distribution between ($r_j + p_j, r_j + p_j + k_j$). Here, *k j* denotes the allowance value, and its range is generated at random from a discrete uniform distribution between [10, 30].

A computational experiment was performed to demonstrate that a solution is searched in an efficient manner by applying the above conditions. The genetic algorithm is programmed in C language and a computer experiment was performed on many problems generated at random.

In order to validate the effectiveness of the proposed genetic algorithm, problems that have 10 jobs($n = 10$) for different parameter sets (p_c, p_m) were generated 10 times at random in order to optimize the performance of the evaluation function, and the experiment was performed on a case where there are 20 parameter sets.

Table 1 shows the types of parameter sets used in the experiment and the results. Here, the average number of generations represents the average number of generations until the optimal solution that the proposed genetic algorithm obtained through enumeration is generated, whereas the average CPU time denotes the average time it took to reach the optimal solution. The maximum number of generations, which was used to optimize the parameter sets, is 500, while the size of the population (pop_size) is 100. As shown Table 1, the optimal solution can be obtained in the shortest time on average when the parameter set (p_c, p_m) has a combination of 0.20 and 0.70. An analysis of this result indicates that mutation contributes more to enhancing the quality of the solution than crossover. It also shows that the higher the ratio for mutation becomes, the less time it relatively takes for searching. Therefore, for the experiment in this study, the fittest parameter set (p_c, p_m) is set to be 0.20 and 0.70 respectively, and the problems are generated 10 times for each of the cases where the numbers of jobs are 6, 7, 8, 9, 10, 15, 20, 30, 40, and 50, and $\max\_gen$ and pop_size are set to be 1,000 and 100 respectively.

Table 1 Results of experiment on parameter set (*Pc* ,*Pm*)

NO.	*Pc*	*Pm*	Avg. CPU time (unit : sec)	Avg. # of replication
1	0.00	1.00	3.40	119.8
2	0.05	0.85	3.80	124.2
3	0.15	0.85	4.40	141.1
4	0.20	0.70	3.00	97.8
5	0.20	0.60	3.30	116.5
6	0.30	0.80	4.90	165.7
7	0.30	0.70	4.34	143.4
8	0.40	0.70	5.42	180.3
9	0.40	0.60	4.09	158.2
10	0.50	0.50	8.17	265.2
11	0.60	0.50	5.45	164.7
12	0.60	0.40	5.60	179.5
13	0.70	0.60	6.28	189.5
14	0.70	0.40	7.10	227.3
15	0.80	0.20	7.35	223.6
16	0.85	0.15	8.43	247.5
17	0.85	0.05	10.82	327.9
18	0.90	0.30	10.50	316.8
19	0.90	0.10	9.84	303.8
20	1.00	0.00	8.47	247.5

In order to evaluate the proposed genetic algorithm, the optimal solution for the cases where the number of job is between 6 and 10 was obtained through the enumeration method and compared against the solution obtained through the genetic algorithm. However, as it was difficult to obtain the optimal solution through the enumeration method for the cases where the number of jobs is over 15, it was compared with the solution obtained through a Tabu search employed by Wan and Yen (2002). The parameters applied to the Tabu search for the same problem generated by the genetic algorithm are the minimum Tabu list whose size is 20, and the size of the maximum Tabu list is three times that of the minimum Tabu list. In addition, the aspiration level is (1.0, 2.0), and the maximum number of iteration varies depending on the size of the job in order to generate the same number of entities as GA.

Table 2 shows a summary of the results of an experiment in which 10 problems were generated at random for each job. Here, * denotes that an optimal solution cannot be obtained within the polynomial time through the enumeration method. OPT is defined as the optimal solution obtained through the enumeration method, while *PGA* is a solution obtained through the genetic algorithm. *TS* is defined as a solution obtained through a Tabu search. The gap between OPT and GA is calculated by $\frac{(OPT - PGA)}{PGA} \times 100$, and the gap between TS and GA is calculated by $\frac{(TS - PGA)}{PGA} \times 100$. The average implies the average for each gap within the size

of a given job. Minimum and maximum mean the minimum gap and the maximum gap, respectively, among the gaps within the above range.

Table 2: Comparison of gap results of GA, OPT and TS (unit: %)

# of Jobs / GAP		6	7	8	9	10	15	20	30	40	50
OPT	Min	0.00	0.00	0.00	0.00	0.00	*	*	*	*	*
vs.	Avg.	0.00	0.00	0.00	0.00	0.00	*	*	*	*	*
PGA	Max	0.00	0.00	0.00	0.00	0.00	*	*	*	*	*
TS	Min	0.00	0.00	0.00	0.00	0.00	0.00	-0.27	-0.22	0.23	0.48
vs.	Avg.	0.00	0.00	0.00	0.00	0.00	0.00	0.11	0.89	1.24	2.67
PGA	Max	0.00	0.00	0.00	0.00	0.00	0.00	0.90	1.61	2.50	6.10

The above results indicate that the suggested genetic algorithm can lead to an optimal solution when the size of the problem is small and that it yields the same solution as or a superior solution to the solution obtained from a Tabu search as the size of the problem increases. This demonstrates the validity of the solution obtained from the proposed genetic algorithm.

CONCLUSIONS

This paper formalized the issue of scheduling for a single machine in consideration of different due dates and arrival times with arithmetic model, and developed a genetic algorithm to find a realizable solution that simultaneously considers arrival times and due dates. To validate the developed genetic algorithm, small-scale problems were compared against the optimal solution obtained through the enumeration method, and large-scale problems were compared against the solution obtained through a Tabu search. The comparisons revealed that the proposed genetic algorithm offers the optimal solution or a similar optimal solution for scheduling problems for a single machine in consideration of different due dates and arrival times.

ACKNOWLEDGEMENTS

The work reported in this paper was conducted during the sabbatical year of Kwangwoon University in 2016.

REFERENCE

[1] Abdul-Razaq, T. S., & Potts, C. N. (1988). Dynamic Programming State-Space Relaxation for Single Machine Scheduling, *Journal of Operational Research Society, 39(2)* 141-152.

[2] Adamopoulos, G. I., & Pappis, C. P. (1996). Scheduling Jobs with Different Job-dependent Earliness and Tardiness Penalties Using the SLK Method, *European Journal of Operational Research*, *88*, 336-344.

[3] Ahmed, M. U., & Sundararaghavan P. S. (1990). Minimizing the Weighted Sum of Late and Early Completion Penalties in a Single Machine, *IIE Transactions*, *22*, 288-290.

[4] Bagchi, U., Sullivan R. S., & Chang Y. L. (1987). Minimizing Mean-squared Deviation of Completion Times about a Common Due Date, *Management Science*, *33*, 894-906.

[5] Baker, K. R., & Scudder, G. D. (1990). Sequencing with Earliness and Tardiness Penalties: a Review, *Operations Research*, *38*, 22-36.

[6] De, P., Ghosh, J. B., & Wells, C. E. (1991). Scheduling to Minimize Weighted Earliness and Tardiness about a Common Due Date, *Computers and Operation Research*, *18*, 463-475.

[7] Fry, T. E., Armstrong, R. D., & Blackstone, J. H. (1987). Minimizing Weighted Absolute Deviation in Single Machine Scheduling, *IIE Transactions*, *19*,

445-450.

[8] Fry, T. D., Armstrong, R. D., & Rosen, L. D. (1990). Single Machine Scheduling to Minimize Mean Absolute Lateness: a Heuristic Solution, *Computers and Operation Research, 17*, 105-112.

[9] Garey, M. R. & Johnson, D. S. (1979). *Computers and Intractability: a guide to the theory of NPcompleteness*, W. H. Freeman, San Francisco, CA

[10] Garey, M.R., Tarjan, R. E., & Wilfong, G. T. (1988). One-processor Scheduling with Symmetric Earliness and Tardiness Penalties, *Mathematics of Operations Research*, 13(2), 330-348.

[11] Glover, F. (1989). Tabu Search-Part I, *ORSA Journal on Computing*, 1, 190-206.

[12] Glover, F. (1989). Tabu Search-Part II, *ORSA Journal on Computing*, 2, 4 –32.

[13] Gupta, Y.P., Gupta, M.C., & Kumar A. (1993). Minimizing Flow Time Variance in a Single Machine System Using Genetic Algorithms, *European Journal of Operational Research*, 70, 289-303.

[14] Hall, N. G., & Posner, M. E. (1991). Earliness-tardiness Scheduling Problems I : Weighted Deviation of Completion Times about a Common Due-date, *Operations Research*, 39. 836-849.

[15] Kanet, J. J. (1981). Minimizing the Average Deviation of Job Completion Times about a Common Due Date, *Naval Research Logistics Quarterly*, 28, 643-651.

[16] Kim, Y.D. & Yano, C.A. (1994). Minimizing Mean Tardiness and Earliness in Single-machine Scheduling Problems with Unequal Due Dates, *Naval Research Logistics*, 41, 913-933.

[17] Lee, C. Y. & Choi, J. Y. (1995). A Genetic Algorithm for Job Sequencing Problems with Distinct Due Dates and General Early-tardy Penalty Weights", *Computers and Operation Research*, 22, 857-869.

[18] Lee, C. Y., Danusaputro, S. L., & Lin, C. S. (1991). Minimizing Weighted Number of Tardy Jobs and Weighted Earliness-tardiness Penalties about a Common Due Date, *Computers and Operation Research*, 18.

[19] Ow, P. S. & Morton, T. E. (1989). The Single Machine Early-Tardy Problem, *Management Science*, 35, 177-191.

[20] Sridharan, V. Zhou, and Z. (1996) Dynamic Non-preemptive Single Machine Scheduling, *Computers and Operation Research*, 23(12), 1183-1190.

[21] Sule, D.R. (1997). *Industrial Scheduling*, PWS Publishing Co.

[22] Yano, C. A. & Kim Y. D. (1991). Algorithms for a Class of Single Machine Tardiness and Earliness Problems, *European Journal of Operational research*, 52, 167-178.

[23] Sourd, F., "Earliness-tardiness scheduling with setup considerations", Computers and Operations Research 32, 1849-1865. 2005.

[24] Jeong, S. J., & Kim, K. S. (2008). Parallel Machine scheduling with earliness-tardiness penalties and space limits, International Journal of Advanced Manufacturing Technology, 37, 793-802.

[25] Wan, . & Yen B. P. –C (2002). Tabu search for single machine scheduling with distinct due windows and weighted earliness/tardiness penalties, European Journal of Operational Research, 142, 271-281.

[26] Sadykov, R. (2008). A branch-and-check algorithm for minimizing the weighted number of late jobs on a single machine with release dates, *European Journal of Operational Research*, 189, 1294-1304.

[27] Valente, J. M. S., & Goncalves J. F. (2009). A genetic algorithm approach for the single machine scheduling problem with linear earliness and quadratic tardiness penalties, *Computers & Operations Research*, 36, 2707-2715.

[28] Pan, J. C. –H., Chen, J. –S., & Cheng, H. –L. (2001). A heuristic approach for single-machine scheduling with due dates and class setups, *Computers & Operations Research*, 28, 1111-1130.

[29] Mazzini, R., & Armentano, V. A. (2001). A heuristic for single machine scheduling with early and tardy costs, *European Journal of Operational Research*, 128. 129-146.

[30] Low, C., H, C –J., & Su, C. –T. (2010). *Expert Systems with Applications*, 37, 6429-6434.

[31] Valente, J. M. S., & Alves, R. A. F. S. (2005). An exact approach to early/tardy scheduling with release dates, *Computers & Operations Research*, 32, 2905-2917.

[32] Soroush, H. M. (2007). Minimizing the weighted number of early and tardy jobs in a stochastic single machine scheduling problem, *European Journal of Operational Research*, 181, 266-287,