Neural Comput & Applic (2015) 26:1439–1450
DOI 10.1007/s00521-014-1789-4

ORIGINAL ARTICLE

A new neuro-dominance rule for single-machine tardiness problem with double due date

Tarik Cakar · Raşit Köker · Ozkan Canay

Received: 2 April 2014 / Accepted: 8 December 2014 / Published online: 6 January 2015

Abstract In this study, the single-machine total weighted tardiness scheduling problem with double due date has been addressed. The neuro-dominance rule (NDR-D) is proposed to decrease the total weighted tardiness (TWT) for the double due date. To obtain NDR-D, a back-propagation artificial neural network was trained using 12,000 data items and tested using another 15,000 items. The adjusted pairwise interchange method was used to prepare training and test data of the neural network. It was proved that if there is any sequence violating the proposed NDR-D then, according to the TWT criterion, these violating jobs are switched. The proposed NDR was compared with a number of generated heuristics. However, all of the used heuristics were generated for double due date based on using the original heuristic (ATC, COVERT, SPT, LPT, EDD, WDD, WSPT and WPD). These generated competing heuristics were called ATC1, ATC2, ATC3, COV1, COV2, COV3, COV4, EDD1, EDD2, EDD3, WDD1, WDD2, WDD3, WSPT1, WSPT2, WSPT3, WPD1, WPD2, WPD3 and WPD4. The arrangements among the heuristics were made according to the double due date. The proposed NDR-D was applied to the generated heuristics and metaheuristics, simulated annealing and genetic algorithms, for a set of randomly generated problems. Problem sizes were chosen as 50, 70 and 100. In this study, 202,500 problems were randomly generated and used to demonstrate the performance of NDR-D. From the computational results, it can be clearly seen that the NDR-D dominates the generated heuristics and metaheuristics in all runs. Additionally, it is possible to see which heuristics are the best for the double due date single-machine TWT problems.

Keywords Single-machine scheduling · Total weighted tardiness · Neuro-dominance rule · Double due date

T. Cakar (✉)
Engineering Faculty, Department of Industrial Engineering, Sakarya University, 54187 Adapazarı, Turkey
e-mail: tcakar@sakarya.edu.tr

R. Köker
Technology Faculty, Department of Electrical and Electronics Engineering, Sakarya University, 54187 Adapazarı, Turkey

O. Canay
Engineering Faculty, Department of Computer Engineering, Sakarya University, 54187 Adapazarı, Turkey

1 Introduction

Companies need to place much emphasis on coordinating priorities through functional fields in order to survive in a strongly competitive commercial environment. The new neuro-dominance rule (NDR-D) provides sufficient conditions for local optimality for a single-machine total weighted tardiness (TWT) problem. The single-machine TWT problem is presented as $1||\sum w_i T_i$. The literature survey for this paper has primarily concentrated on single-machine TWT and the same problem with due date. Subsequently, the same problem solved using artificial intelligence methods was reviewed and reported in this paper. Hsu et al. [1] focused on the analysis of single-machine scheduling and due date assignment problems based on position-dependent processing time. In their paper, two frequent due date assignment methods and two generally positional deterioration models are presented. The target functions include the cost of changing the due dates, the total cost of positional weight earliness and the total cost of discarded jobs which cannot be completed by their due dates. In another study by Gordon and Strusevich [2], single-machine scheduling and due date assignment problems with position-dependent processing time were investigated. Shabtay and Steiner [3] published a paper on two single-machine scheduling problems based on the minimization of the sum of weighted earliness, tardiness and due date assignment penalties and minimization of the weighted number of tardy jobs and due date assignment costs. Wang [4] presents a study on an iterative bidding framework for integrated due date management decision-making.

Lawler [11] defined the TWT problem strongly as $1||\Sigma w_i T_i$ and gave a pseudo-polynomial algorithm for the total tardiness problem, $\Sigma w_i T_i$. For weighted and unweighted tardiness problems, a few different solutions have been given in [12, 13, 14]. Emmons's study [13] is based on deriving several superior rules, which limits the search for an optimum solution to the $1||\Sigma w_i T_i$ problem. The rules mentioned in the studies are used in both branch-and-bound (B&B) and dynamic programming algorithms (Fisher [15] and Potts and Van Vassenhove [16, 17]). Rinnooy Kan et al. [14] extended these results to the weighted tardiness problem. The significance of a customer depends on various factors, but it is important in manufacturing to reflect these priorities in scheduling decisions. For this reason, a new superior rule for the most general case of the TWT problem has been presented by the authors. Our suggested rule improves and also covers Emmon's results and the generalizations of Rinnooy Kan et al. under the evaluation of the time-dependent orderings between each pair of jobs. Since implicit enumerative algorithms may require important computer resources, both in terms of memory and computational time, various heuristics and dispatching rules have been proposed. These studies are significant since they have led to the discovery of dominance and neuro-dominance rules.

Chambers et al. [12] published a paper based on the improvement of novel heuristic superior rules and a flexible decomposition heuristic. Abdul-Razaq et al. [19] tested the exact approaches used to solve the weighted tardiness problem, and Emmons's superior rules have been used in their paper to form a precedence graph for finding upper and lower bounds. They reported that the most promising lower bound, both in time consumption and quality, is Potts and Van Wassenhove's [16] linear lower bound method, obtained from Lagrangian slacking of the machine capacity constraint.

The problem of scheduling n jobs with release dates, due dates, weights and equal process times on a single machine has been studied by Akker et al. [5]. The target is the minimization of TWT. Li et al. [6] presented a paper based on the investigation of how to sequence jobs with fuzzy processing times and to predict their due dates on a single machine such that the total weighted possibilistic mean value of the weighted earliness–tardiness costs is minimized. Kellegoz et al. [7] presented a paper comparing the efficiencies of genetic crossover operators for the one-machine TWT problem. Yoon and Lee [8] proposed a new heuristic for the single-machine TWT problem. Three new heuristic algorithms have been proposed and compared with other competing heuristics from the literature. Even if most practical problems deal with multiple resources, many problems can be broken down into a subsequently solved series of single-machine problems, as, for example, in the famous shifting bottleneck heuristic [9]. Colka et al. [10] suggested an interval-indexed formulation-based heuristic for the single-machine TWT problem.

A new neural network approach for solving the single-machine mean tardiness scheduling problem and the minimum makespan job-shop scheduling problem has been suggested by Sabuncuoglu and Gurgun [18]. Weckman et al. [37] suggested a neural network scheduler to be used in job-shop scheduling. Dudek-Dyduch [35] presented a paper on considering intelligent-learning-based algorithms in scheduling problems. Laguna et al. [36] presented a paper on the discussion of the use of three local search strategies within a tabu search method for the approximate solution of a single-machine scheduling problem. Yim and Lee [38] suggested a new method to schedule cluster tools in semiconductor production. A real-time fuzzy expert system to schedule parts for a flexible manufacturing system was suggested by Chan et al. [34]. Bozejko [20] dealt with the parallel path relinking method for the single-machine TWT problem with sequence-dependent setups.

In their paper, Manham and Moslehi [22] studied the problem of the minimization of the sum of maximum earliness and tardiness on a single machine with unequal release times. It has been proven that this problem is NP-hard in the strong sense and a B&B algorithm was developed as an exact method. Li et al. [23] dealt with the single-machine scheduling problem for the minimization of total resource consumption under the constraint that the makespan does not exceed a given limit, in which the release date of a job is a linear decreasing continuous function of the resource consumption. Greiger [24] published a paper about the theoretical and experimental investigations of computational intelligence techniques regarding machine sequencing problems. Eren [25] considered a single-machine scheduling problem with unequal release dates and a learning effect. The target function of the problem is to minimize the total weighted completion time. A nonlinear mathematical programming model has been proposed to obtain an exact solution to the problem. Baker and Keller [26] presented a paper comparing six different integer programming formulations of the single-machine total tardiness problem.

In a study presented by Vepsalainen and Morton [27], efficient dispatching rules were developed and tested. By their proposed superior rule, an adequate condition for local optimality was provided, and schedules, which cannot be developed by adjacent job interchanges, were generated from it. In another paper, more practical applications of the weighted tardiness problem and computing the lower bound were presented by Akturk and Yildirim [28]. Dominance conditions proved to be especially useful in the reduction in the size of problems when scheduling jobs on a single machine for the minimization of the weighted total tardiness [29]. A dominance rule has been informally defined by Kanet [30] as identifying a subset of solutions, which contain at least one optimal solution for a problem. According to Kanet's scheduling context [29], a dominance condition is described as a rule, which specifies that one job will precede another if certain conditions hold. A bi-criteria scheduling problem, in which two target functions are maximum lateness induced by two sets of due dates, has been studied in another paper by Cheng et al. [21].

A new dominance rule for the $1|rj|\Sigma w_i T_i$ problem, which can be used in reducing the number of alternatives in any exact approach, was presented by Akturk and Ozdemir [31]. An interchange function, used by Akturk and Yildirim, $\Delta ij(t)$, has been used to specify the new dominance properties that give the cost of interchanging adjacent jobs i and j whose processing starts at time t. Akturk and Yildirim reported that they found three breakpoints by using cost functions and obtained a number of rules by using the breakpoints. Cakar [32, 33] proposed a neuro-dominance rule for the single-machine tardiness problem without release dates and also a neuro-dominance rule for the single-machine tardiness problem with unequal release dates. During the study, job sizes of $n = 50$, 70, 100 and six different heuristics were implemented.

In this paper, a back-propagation artificial neural network (BPANN) has been trained to show how the proposed superior rule can be used for the development of a sequence that is given by a dispatching rule. We also present the proof if any sequence disturbs the proposed superior rule, the switching of the disturbing jobs either lowers the TWT or leaves it unchanged. According to the literature, due to its exhaustive computational needs, the weighted tardiness problem is NP-hard and the lower bounds do not have practical applications.

In this paper, instead of extracting the rules to find the break point using cost functions, a neural network was trained by using a sufficient number of data, different from those of Aktürk and Yıldırım [28] and Akturk and Ozdemir [31]. In the event of giving the necessary inputs, which job will come first among the adjacent jobs was decided according to the TWT problem criterion. To emphasize another different point of the proposed study, there are double due dates for each job in this study. The studies presented by Akturk, Cakar and Hsu [1] are closer to this study. Only Hsu studied the double due date and made the double due date an assignment. On the other hand, in Akturk's studies [28, 31], the dominance rules have been ascertained, but single due date has been used. Additionally, during Cakar's studies, the neuro-dominance rule was used, but the study was undertaken for a single due date.

This paper is organized as follows. In Sect. 2, the parameters used in the problems, modeling of the problem and the working principle of NDR-D are given. In Sect. 3, the generated heuristics are explained. In Sect. 4, genetic algorithms and the simulated annealing algorithm are compared. Finally, in Sect. 5, all the computational results and analyses are given.

2 Definition of the problem

The single-machine problem with double due date can be explained as follows. Each job that is numbered from 1 to n is processed without any interruption on a single machine that can undertake only one job at a time. If i represents a job, it has five parameters that are p_i, d_{i1}, d_{i2}, w_{i1} and w_{i2}, which refer to an integer processing time, double due date and double positive weights (tardiness costs), respectively. We can define the problem as finding the schedule K based on the minimization of the $f(k)$ function. In our selected problem, when the due date is passed, there is a double due date; according to the passed time, two different delay costs are applied. The TWT calculation is presented below:

$$f(S) = \begin{cases} \text{If } C_i > d_{i2} \quad \text{then TWT} = \sum \left[(d_{i2} - d_{i1}) w_{i1} + (C_i - d_{i2}) w_{i2} \right] \\ \text{If } d_{i1} < C_i \le d_{i2} \quad \text{then TWT} = \sum (C_i - d_{i1}) w_{i1} \\ \text{If } C_i \le d_{i1} \quad \text{then TWT} = 0 \end{cases}$$

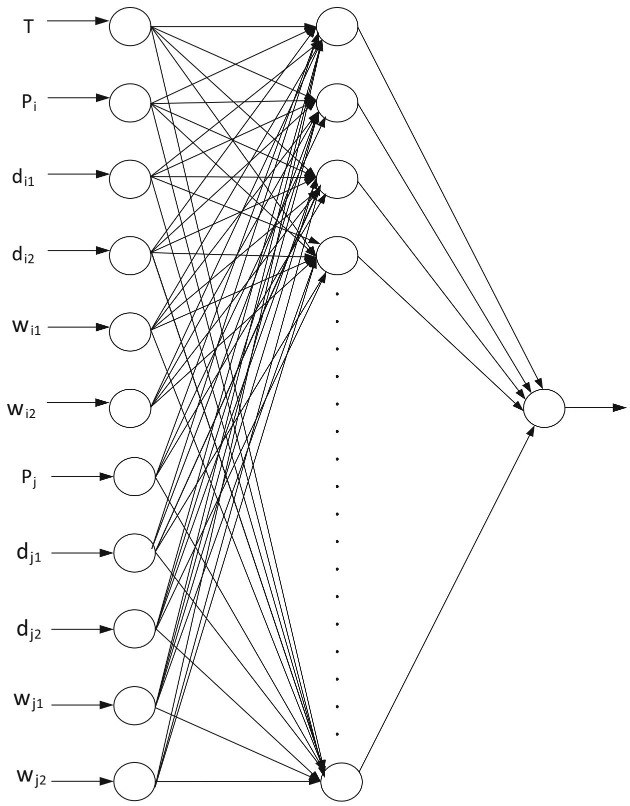

Fig. 1 Structure of the used BPANN 11 inputs and one output

The neuro-dominance rule can be introduced by considering two schedules $K_1 = L_1 ijL_2$ and $K_2 = \mathrm{L}_1 jiL_2$, where L_1 and L_2 are two disjoint subsequences of $n-2$ remaining jobs, which are separated by i and j. It should be noted that the completion time of L_1 is $C = \sum_{k \in L1} p_K$.

In this paper, which job will be done first between two adjacent jobs has been decided based on the TWT criterion using a BPANN. The first and second jobs have been taken as i and j without taking into account the due date or processing time for these two adjacent jobs. The designed neural network consists of 11 inputs and one output and 30 neurons in the hidden layer. The inputs of the BPANN are the starting time of job i (T), the processing time of job i (p_i), the due dates of job i (d_{i1},d_{i2}), the weights of job i (w_{i1}, w_{i2}), the processing time of job j (pj), the due dates of job j (d_{j1},d_{j2}), and the weights of job j (w_{j1}, w_{j2}). Values "0"and "1" are used for the determination of the precedence of the jobs. If the obtained output value from the BPANN is equal to "0", then i should precede j. The topology of the designed BPANN is given in Fig. 1. The training and testing parameters of the designed neural network are also given in Table 1. By using Fig. 2 and the data presented in Table 2, how the NDR-D works can be seen. The decision mechanism of the suggested neuro-dominance rule is presented in Fig. 2. The adjusted pairwise interchange (API) method has been used to prepare the data for the training and testing of the neural network. The proposed study is presented in Fig. 3 as a block scheme.

The single-machine total tardiness problem with double due date is a study which is practically applied in industry. In particular, in the event that a company, which is a supplier, cannot deliver products on time, the company

Table 1 Training and test parameters of the artificial neural networks

Sample size in training set	12.000
Achievement rate of the training data	%100
Number of test data	15.000
Achievement rate of the test data	%100
Activation function	Sigmoidal
Iteration number	6.000.000
Learning rate	0.38
Momentum rate	0.72

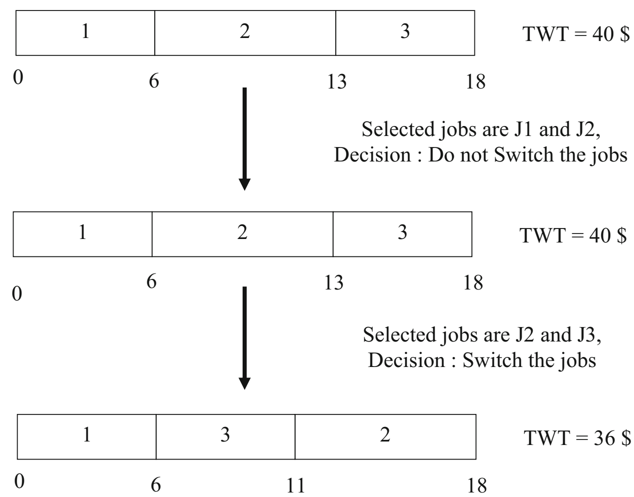

Fig. 2 Decision mechanism of the proposed neuro-dominance rule

Table 2 p, d_1, d_2, w_1 and w_2 for example problem

Job	Definition	p (day)	d_1 (day)	d_2 (day)	w_1($)	w_2 ($)
1	Cutting	6	5	7	1	2
2	Grinding	7	9	11	2	4
3	Lathing	5	10	15	3	4

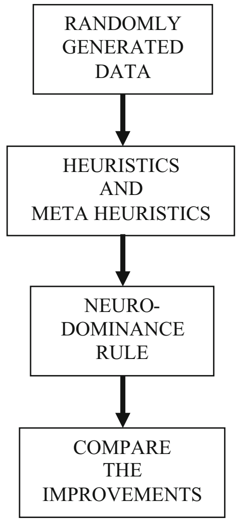

Fig. 3 Structure of the proposed study

using the products cannot deliver its products on time and is exposed to financial penalties. Therefore, if the supplier company cannot deliver the product on the first delivery date, it waits for the second delivery date to deliver the product and in this case, the company has to pay higher delay penalties. The supplier company and the customer previously agreed on the delivery date and their unit delay penalty. Examples of large companies working with this double due date, are Turkish Coach Industry Incorporation (TUVASAS) and ELIMSAN A.S. These companies apply the double due date method in delivering some parts. Generally, these parts are those which significantly affect on-time delivery (first degree). Therefore, companies cannot deliver products on time and they are exposed to delay penalties. Thus, they apply the delay penalties to the suppliers of this type of part. There are no technological precedence constraints between jobs. p is processing time, d_1 and d_2 are due dates, and w_1 and w_2 are unit tardiness costs.

3 Generated heuristics for double due date

Different heuristics versions such as COVERT, ATC, EDD, SPT, LPT, WDD, WSPT and WPD have been generated. By applying NDR-D to these generated heuristics, the success of NDR-D can be seen and the successful proposed heuristics for double due date can be determined. The proposed heuristics for double due date are as given below:

3.1 SPT

SPT1 $\min[p_i]$

3.2 LPT

LPT $\max[p_i]$

3.3 EDD

EDD1 $\min\left[d_i^1\right]$

EDD2 $\min\left[d_i^2\right]$

EDD3 $= \min[d_i]$ where $d_i = \left\{ \begin{array}{l} \min\left[d_i^2\right], t \geq d_i^2 \\ \min\left[d_i^1\right], t < d_i^2 \end{array} \right\}$

3.4 WDD

WDD1 $\max\left[\frac{\mathrm{w_i^1}}{\mathrm{d_i^1}}\right]$

*WDD*2 $\max\left[\frac{w_i^2}{d_i^2}\right]$

WDD3 $\max\left[\frac{w_i^1 + w_i^2}{d_i^1 + d_i^2}\right]$

3.5 WSPT

WSPT1 $\max\left[\frac{w_i^1}{p_i}\right]$

WSPT2 $\max\left[\frac{w_i^2}{p_i}\right]$

WSPT3 $\max\left[\frac{w_i^1 + w_i^2}{p_i}\right]$

3.6 WPD

WPD1 $\max\left[\frac{w_i^1}{p_i d_i^1}\right]$

WPD2 $\max\left[\frac{w_i^2}{p_i d_i^2}\right]$

WPD3 $\max\left[\frac{w_i^1 + w_i^2}{p_i(d_i^1 + d_i^2)}\right]$

WPD4 $\max\left[\frac{w_i^1}{p_i d_i^1} + \frac{w_i^2}{p_i d_i^2}\right]$

3.7 ATC

$$\mathrm{ATC}(1) = \max\left[\frac{w_i^1 + w_i^2}{2p_i} \exp\left(-\frac{\max\left(0, \frac{\mathrm{d_i^1 + d_i^2}}{2} - \mathrm{t} - \mathrm{p_i}\right)}{k\overline{p}}\right)\right]$$

$$\mathrm{ATC}(2) = \max\left[\frac{w_i^1 + w_i^2}{2p_i} \exp\left(0, 1 - \frac{\max\left(0, \overline{\mathrm{d}}_\mathrm{i} - \mathrm{t} - \mathrm{p_i}\right)}{k\overline{p_i}}\right)\right]$$

where $\overline{d_i} = \frac{w_i^1 d_i^1 + w_i^2 d_i^2}{w_i^1 + w_i^2}$

$$ATC(3) = \left\{ \begin{array}{l} \max\left[\frac{w_i^1}{p_i}\exp\left(-\frac{\max(0, \mathrm{d}_i^1 - \mathrm{t} - \mathrm{p}_i)}{k\overline{p}}\right)\right], t \le d_i^1 \\ \max\left[\frac{w_i^1}{p_i}\exp\left(-\frac{\max(0, \mathrm{d}_i^1 - \mathrm{t} - \mathrm{p}_i)}{k\overline{p}}\right)\right] + \max\left[\frac{w_i^2 - w_i^1}{p_i}\exp\left(0, 1 - \frac{\max(0, \mathrm{d}_i^2 - \mathrm{t} - \mathrm{p}_i)}{k\overline{p}}\right)\right], t\rangle d_i^1 \end{array} \right\}$$

3.8 COVERT

$$\mathrm{COVERT}(1) = \max\left[\frac{w_i^1 + w_i^2}{2p_i}\max\left(0, 1 - \frac{\max\left(0, \frac{\mathrm{d}_i^1 + \mathrm{d}_i^2}{2} - \mathrm{t} - \mathrm{p}_i\right)}{kp_i}\right)\right]$$

$$\mathrm{COVERT}(2) = \max\left[\frac{w_i^1 + w_i^2}{2p_i}\max\left(0, 1 - \frac{\max(0, \overline{\mathrm{d}_i} - \mathrm{t} - \mathrm{p}_i)}{kp_i}\right)\right]$$

$$\text{where } \overline{d_i} = \frac{w_i^1 d_i^1 + w_i^2 d_i^2}{w_i^1 + w_i^2}$$

$$\mathrm{COVERT}(3) = \max\left[\frac{w_i^1}{p_i}\max\left(0, 1 - \frac{\max(0, \mathrm{d}_i^1 - \mathrm{t} - \mathrm{p}_i)}{kp_i}\right) + \frac{w_i^2}{p_i}\max\left(0, 1 - \frac{\max(0, \mathrm{d}_i^2 - \mathrm{t} - \mathrm{p}_i)}{kp_i}\right)\right]$$

$$\mathrm{COVERT}(4) = \max\left[\frac{w_i^1}{p_i}\max\left(0, 1 - \frac{\max(0, \mathrm{d}_i^1 - \mathrm{t} - \mathrm{p}_i)}{kp_i}\right) + \frac{w_i^2 - w_i^1}{p_i}\max\left(0, 1 - \frac{\max(0, \mathrm{d}_i^2 - \mathrm{t} - \mathrm{p}_i)}{kp_i}\right)\right]$$

4 Simulated annealing

The SA method, based on the physical phenomenon of annealing, is a metaheuristic technique to solve combinatorial optimization problems. It was first suggested in the literature by Kirkpatrick et al. [39] in 1983. A combinatorial optimization problem is solved by using the SA in a manner which is analogous to the process of annealing. The method is based on two results from statistical physics: the probability of a system achieving a given energy E at thermodynamic balance and the so-called Metropolis algorithm that may be used in the simulation of the evolution of a system toward thermodynamic balance at a given temperature. To mimic the temperature of the system, a control parameter is introduced. The number of accessible energy states is controlled by the temperature, and the temperature leads to a locally optimal state in the event of gradual lowering. In the system, the energy resembles the objective function value in a minimization problem, while a feasible solution resembles a certain state of the system. The final solution resembles the system becoming frozen in its ground state [40].

Representation of the SA algorithm as a flowchart is given in Fig. 4 [41]. On the other hand, the pseudo-code of the SA is given below:

```
Start;
   Initialize (A, C, T);
Repeat
   For I = 1 to C do
      N = Perturb (A); (Generate new neighborhood solution)
      D = C(N)−C(A)
      If C(N) <=C(A) or
(exp (−D/T) > Random(0,1))
      Then A = N; (Accept the movement)
     Endif
   Endfor
Until (Stopping Criteria satisfied)
Stop;
```

As can be seen in Fig. 4, the SA starts with an initial solution (A), initial temperature (T) and iteration number (C). The temperature controls the possibility of the acceptance of the disturbing solution as mentioned above. However, the reason for using the iteration number is to decide the number of repetitions until a solution is found in a stable state under the temperature [42, 43]. The temperature achieves the following implicit flexibility index meaning. At the beginning of the searches, at a high-temperature situation, some flexibility may move to a worse case solution; on the other hand, less of this flexibility exists in the searches undertaken later, meaning at a lower temperature. A new neighborhood solution (N) will be generated based on this T and C through heuristic perturbation to the existing solutions. When an improvement has been observed on changing an objective function, the neighborhood solution (N) will be seen as a

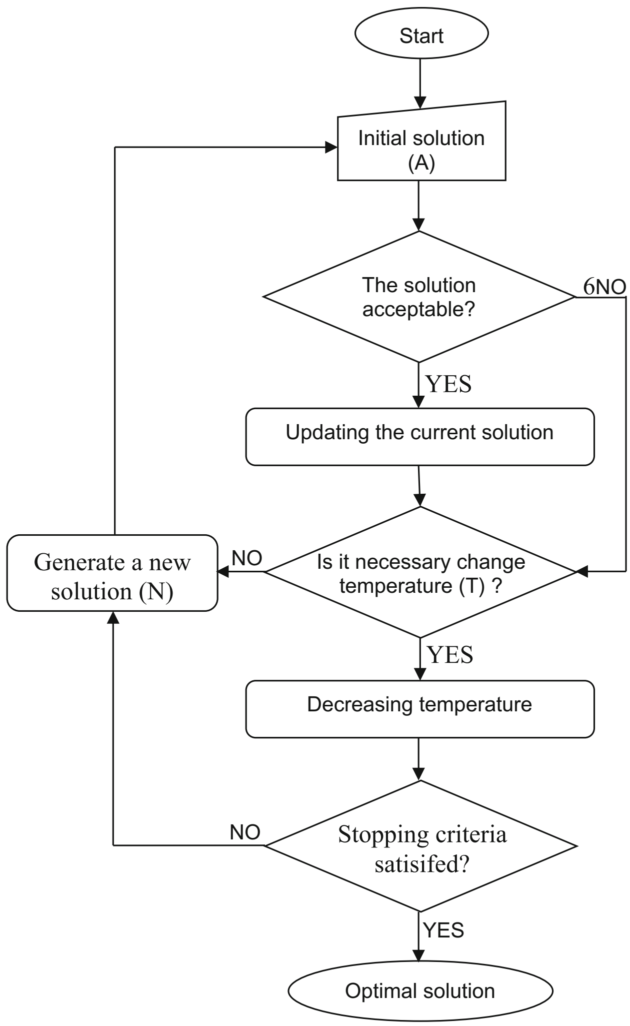

Fig. 4 Flowchart and pseudo-code of the simulating algorithm

good solution. Even if the change in an objective function is not improved, the neighborhood solution will be a new solution with a convenient probability, which is based on $e^{-D/T}$. This situation removes the possibility of finding a global optimum solution out of a local optimum. If there is no change after certain iterations, the algorithm will be stopped. The algorithm carries on with a new temperature value if there is still improvement in the new solution.

Table 3 Experimental design

Factors	Distribution range
Number of jobs	50,70,100
Processing time range	[1–10], [1–50], [1–100]
Weight range (w_{i1})	[1–10], [1–50], [1–100]
Weight range (w_{i2})	[1–10], [1–50], [1–100]
RDD	0.1, 0.3, 0.5, 0.7, 0.9
TF	0.1, 0.3, 0.5, 0.7, 0.9

5 Computational results

In this study, a set of randomly generated problems was used to test the TWT improvement by using NDR-D. The TWT improvement was tested by the authors on problems generating 50, 70 and 100 jobs. For each job, i, p_i, w_{i1} and w_{i2} have been generated from three uniform distributions, [1, 10], [1, 50] and [1, 100] to make up low, medium or high variations, respectively. Here, as mentioned earlier, p_i, w_{i1} and w_{i2} represent an integer processing time and integer weights, respectively. The proportional range of the due dates (RDD) and the average tardiness factor (TFF) have been selected from the set {0.1, 0.3, 0.5, 0.7, 0.9}. d_{i1}, an integer due date from the distribution [$P(1 - \mathrm{TF} - \mathrm{RDD}/2)$, $P(1 - \mathrm{TF} + \mathrm{RDD}/2)$] and d_{i2}, an integer second due date from the distribution [d_{i1}, $P(1 - \mathrm{TF} + \mathrm{RDD}/2)$], have been generated for each job i. Here, P represents the total processing time, $\sum_{i=1}^{n} p_i$. As shown in Table 3, the authors examined and evaluated 2,025 example sets and achieved 100 replications for each combination, resulting in 202,500 randomly produced runs.

A number of heuristics have been used to find an initial sequence for TWT. Some heuristics have been adapted to double due dates. Different versions of the same heuristic, adapted according to the double due date, have been generated. The heuristics are SPT, LPT, ATC1, ATC2, ATC3, COV1, COV2, COV3, COV4, EDD1, EDD2, EDD3, WDD1, WDD2, WDD3, WSPT1, WSPT2, WSPT3, WPD1, WPD2, WPD3 and WPD4. The indexes and the formulations of the heuristics have been given above. In their paper, Vepsalainen and Morton [27] discussed the ATC rule as superior to other sequencing heuristics, and they described it as being close to the optimal for the $\sum w_i T_i$ problem.

In this study, SA and genetic algorithms (GAs), two different metaheuristic algorithms, have been used in addition to heuristics. The parameters used and the operators for the implementation of the SA to generate new solutions are described below. A new solution means a new job sequence. During the study, for the generation of a new neighborhood solution, three different operators were used. The operators can be described as swap-1, swap-2 and inverse operators. On the other hand, TWT has been taken as a fitness function. During the implementation of SA, the best value, which has been achieved from the heuristics, has been taken as a starting solution.

5.1 Swap operator

Old Solution :	3	5	***1***	7	2	8	4	***9***	6
New Solution :	3	5	***9***	7	2	8	4	***1***	6

5.2 Swap operator

Old Solution :	2	***8***	***9***	3	5	1	4	***6***	***7***
New Solution :	2	***6***	***7***	3	5	1	4	***8***	***9***

5.3 Inverse operator

Old Solution :	*6*	1	***4***	***7***	***3***	***8***	***9***	2	5
New Solution :	6	1	***9***	***8***	***3***	***7***	***4***	2	5

In the swap operators above, the jobs, which will be swapped, have been completely determined randomly. Furthermore, the beginning and end points of the array of the jobs, which will be inversely sequenced, have been randomly determined. The geometric ratio has been applied in SA, and the starting temperature and the cooling ratio have been taken as 12,000 and 0.95, respectively.

GAs do not deal with the problem, but they deal with the code of the problem. Thus, the problem, for which we are looking for a solution in using the GA, has to be coded correctly. After the correct coding process, the application of genetic operators to the coded problem is necessary. Two different genetic operators are used: crossover and mutation operators. These operators are applied to the chromosomes. The initial population consists of the results obtained from priority rules and randomly generated solution alternatives for any problem. TWT has been defined as a fitness function. The population has been taken as 100. The maximum number of generations is 250. Crossover and mutation rates have been taken as 100 and 5 %, respectively.

The linear order crossover (LOX) method has been applied to each chromosome independently. LOX works as follows:

5.4 Select the sublist from chromosomes randomly

Chromosome-1 :	1	2	3	***4***	***5***	***6***	7	8	9
Chromosome-2 :	6	4	5	***7***	***1***	***3***	2	9	8

5.5 Remove the sublist from chromosomes randomly

Chromosome-1 :	**t**	2	**t**	4	5	6	**t**	8	9
Chromosome-2 :	2	4	5	**t**	**t**	**t**	6	9	8

5.6 Remove the sublist 1 from chromosome #2

Chromosome-1 :	t	t	**t**	7	1	3	2	8	9
Chromosome-2 :	2	4	5	t	t	t	6	9	8

5.7 Insert the sublist into holes to form offspring

Offspring-1 :	2	4	5	7	1	3	6	8	9
Offspring-2 :	7	1	3	4	5	6	2	9	8

Working form of mutation operator is as follows: Two different genes are selected from randomly selected chromosome and swapped.

Chromosome :	9	6	***5***	8	1	2	***3***	7	4
Offspring :	9	6	***3***	8	1	2	***5***	7	4

Table 4 Computational results for $n = 50$

Heuristic	Total weighted tardiness		
	Before	After (+NDR-D)	$\overline{imprv}$
SPT	40,281.45	39,137.78	7.2,358921
LPT	120,524.2	114,287.6	3.1247573
EDD1	10,829.4	10,263.21	0.0129086
EDD2	64,238.71	59,875.35	1.9863241
EDD3	59,248.33	57,668.81	4.95214
WDD1	34,529.8	32,721.4	7.17448

Table 4 continued

Heuristic	Total weighted tardiness Before	After (+NDR-D)	$\overline{imprv}$
WDD2	48,027.72	46,210.77	7.65201
WDD3	39,623.02	38,477.36	9.158731
WSPT1	24,661.89	26,093.69	1.67294
WSPT2	31,120.43	31,102.01	0.047547
WSPT3	28,606.21	20,019.9	12.64875
WPD1	28,838.99	28,541.16	0.925481
WPD2	35,996.28	34,642.3	2.235772
WPD3	30,506.94	30,629.1	1.377545
WPD4	29,201.31	29,189.34	4.98647
ATC1	86,395.74	82,589.73	0.24617
ATC2	84,368.27	80,332.96	6.00458
ATC3	33,587.85	33,254.36	1.01354
COV1	91,827.3	88,295.7	4.87365
COV2	89,598.75	86,200.38	10.97063
COV3	33,445.25	30,852.91	4.87322
COV4	36,631.41	35,744.24	9.16974
GA	66,297.88	66,275.12	0.0019
SA	66,687.97	66,481.74	0.0094

Table 5 Computational results for $n = 70$

Heuristic	Total weighted tardiness Before	After (+NDR-D)	$\overline{imprv}$
SPT	119,380.1	115,056.9	1.287512
0LPT	358,266.7	347,897.4	2.468947
EDD1	24,244.0	24,200.3	0.012587
EDD2	195,288.8	186,101.6	12.10651
EDD3	189,112.4	185,722.8	6.470266
WDD1	94,287.2	93,432.17	10.24698
WDD2	132,188.5	126,507.9	1.036236
WDD3	105,801.4	101,638.7	3.825478
WSPT1	62,738.31	59,754.3	3.980559
WSPT2	83,809.05	83,623.45	0.068224
WSPT3	69,995.85	68,251.72	10.00674
WPD1	74,566.36	74,210.72	0.852770
WPD2	98,432.8	96,214.5	0.509264
WPD3	79,147.24	79,002.32	1.589525
WPD4	79,214.81	77,358.74	4.658250
ATC1	210,215.2	203,985.2	0.125873
ATC2	206,878.4	197,914.1	4.346512
ATC3	83,545.36	82,697.81	6.047012
COV1	222,836.6	215,652.7	5.659874
COV2	219,291.3	211,243.2	9.497525
COV3	85,033.57	84,399.75	5.69318
COV4	88,817.8	88,356.41	8.1295
GA	174,724.2	174,535.3	0.0019
SA	174,638.4	174,274.9	0.0093

Table 6 Computational results for $n = 100$

Heuristic	Total weighted tardiness Before	After (+NDR-D)	$\overline{imprv}$
SPT	241,382.7	237,318.0	9.523721
LPT	706,592.4	689,426.2	0.874219
EDD1	37,735.1	35,217.4	0.014367
EDD2	218,378.3	215,994.5	1.995400
EDD3	410,200.7	404,328.2	3.467201
WDD1	216,437.8	212,020.3	9.74902
WDD2	286,804.6	279,387.5	9.782479
WDD3	237,653.4	231,985.9	6.82580
WSPT1	63,328.8	40,824.2	6.81641
WSPT2	186,712.0	186,129.8	0.050247
WSPT3	154,146.6	150,345.7	11.25810
WPD1	166,838.5	165,880.5	0.94995
WPD2	222,873.1	216,388.4	4.28558
WPD3	176,293.7	176,273.0	1.41602
WPD4	178,959.1	170,639.6	4.28759
ATC1	488,327.0	480,733.7	0.114872
ATC2	476,322.2	467,310.4	1.725487
ATC3	196,014.0	195,240.1	3.245421
COV1	542,478.0	533,257.2	7.72458
COV2	530,537.9	519,658.8	8.36799
COV3	203,016.6	198,944.5	7.49763
COV4	216,958.7	214,323.8	6.42154
GA	405,985.0	405,312.1	0.0021
SA	394,257.0	393,987.0	0.0098

NDR-D controls the tandem ordered process order; if there is any change which decreases the TWT among the jobs coming sequentially, the NDR-D provides this. At the beginning, the starting job orders are obtained by using dispatching rules to test NDR-D. Then, NDR-D is applied to these orders and the success of the NDR-D is seen. But, at the same time, the success of these proposed dispatching rules can be seen. Among these dispatching rules, a dispatching rule which improves the NDR-D the least means that it works better than the others from the viewpoint of minimizing the TWT. Here, to see the success rate of the NDR-D, the improvement of each example is found and then the average value of these improvements is computed to obtain average improvement ($\overline{imprv}$). In Tables 4, 5, and 6, average improvement values are given ($\overline{imprv}$). The priority rules improved by NDR-D are EDD1, WSPT2, WPD1 and ATC1. This means that these four priority rules will be the most successful priority rules with regard to minimizing TWT. Additionally, these priority rules have been designed and proposed for double due date.

Table 7 Comparison of TWT after using NDR-D for $n = 50$

Heuristic	$n = 50$			
	Better	Equal	Worse	t
SPT + NDR-D	39,321	28,179	0	17.524
LPT + NDR-D	44,609	22,890	1	5.4589
EDD1 + NDR-D	45,548	21,952	0	6.426
EDD2 + NDR-D	40,376	27,124	0	4.0092
EDD3 + NDR-D	50,552	16,948	0	1.9701
WDD1 + NDR-D	40,773	26,726	1	1.6147
WDD2 + NDR-D	40,203	27,497	0	4.7093
WDD3 + NDR-D	40,580	26,920	0	9.1190
WSPT1 + NDR-D	39,312	28,188	0	24.648
WSPT2 + NDR-D	38,656	28,844	0	33.381
WSPT3 + NDR-D	40,312	27,188	0	15.450
WPD1 + NDR-D	39,726	27,774	0	4.9792
WPD2 + NDR-D	39,834	27,666	0	5.1808
WPD3 + NDR-D	39,192	28,307	1	6.1867
WPD4 + NDR-D	41,338	26,162	0	3.8227
ATC1 + NDR-D	41,155	26,345	0	2.9839
ATC2 + NDR-D	40,537	26,963	0	5.8424
ATC3 + NDR-D	41,424	26,076	0	14.420
COV1 + NDR-D	48,500	19,000	0	16.935
COV2 + NDR-D	47,652	19,848	0	4.1992
COV3 + NDR-D	47,053	20,446	1	26.199
COV4 + NDR-D	47,404	20,096	0	36.006
GA + NDR-D	40,350	27,150	0	9.3915
SA + NDR-D	40,953	26,547	0	26.256

Table 8 Comparison of TWT after using NDR-D for $n = 70$

Heuristic	$n = 70$			
	Better	Equal	Worse	t
SPT + NDR-D	37,851	29,648	1	3.580
LPT + NDR-D	40,464	27,036	0	7.708
EDD1 + NDR-D	46,433	21,067	0	4.265
EDD2 + NDR-D	37,812	29,688	0	2.2683
EDD3 + NDR-D	49,736	17,764	0	4.3226
WDD1 + NDR-D	29,275	38,225	0	9.5216
WDD2 + NDR-D	29,662	37,838	0	9.6981
WDD3 + NDR-D	29,327	38,171	2	17.392
WSPT1 + NDR-D	28,881	38,619	0	29.943
WSPT2 + NDR-D	28,705	38,795	0	35.219
WSPT3 + NDR-D	28,500	39,000	0	26.596
WPD1 + NDR-D	29,246	38,253	1	2.8818
WPD2 + NDR-D	28,349	38,151	0	3.2676
WPD3 + NDR-D	28,352	39,148	0	2.1984
WPD4 + NDR-D	28,923	38,577	0	9.8125
ATC1 + NDR-D	29,344	38,156	0	45.593
ATC2 + NDR-D	29,782	37,717	1	39.499
ATC3 + NDR-D	30,762	36,738	0	18.576
COV1 + NDR-D	39,551	27,749	0	28.853
COV2 + NDR-D	39,529	27,971	0	29.696
COV3 + NDR-D	38,228	29,272	0	20.743
COV4 + NDR-D	38,103	29,396	1	27.931
GA + NDR-D	29,674	37,726	0	34.876
SA + NDR-D	29,406	38,094	0	31.939

SA and GA metaheuristics are more successful than the other heuristics as the best solutions obtained from heuristics are given to SA and the GAs as the initial solutions. Therefore, it is inevitable that they will find better solutions than the others. Each heuristic and metaheuristic, over 67,500 runs for cases of 50, 70 and 100 jobs, has been tested by the authors, and the results are presented in Tables 7, 8 and 9. As can be seen in the tables below, the number of samples which gave the better solution, worse solution and the solution with the same performance level has been presented. The results show that the proposed dominance rule provides an important enhancement to all rules, and the enhancement is noteworthy at the 99.5 % confidence level for all heuristics according to the large t test values on the average enhancement.

6 Conclusion

The authors have presented a new neuro-dominance rule (NDR-D) for the single-machine total tardiness problem with double due date. The proposed algorithm was implemented on a set of heuristics and metaheuristics. After using NDR-D, the heuristics and metaheuristics were compared according to TWT. The results show that some

Table 9 Comparison of TWT after using NDR-D for $n = 100$

Heuristic	$n = 100$			
	Better	Equal	Worse	t
SPT + NDR-D	37,476	30,024	0	2.563
LPT + NDR-D	39,953	27,547	0	6.988
EDD1 + NDR-D	48,748	18,752	0	3.126
EDD2 + NDR-D	48,225	19,274	1	3.3556
EDD3 + NDR-D	50,638	16,862	0	7.3536
WDD1 + NDR-D	35,627	31,873	0	14.210
WDD2 + NDR-D	38,324	29,175	1	12.705
WDD3 + NDR-D	38,363	29,137	0	19.116
WSPT1 + NDR-D	36,512	30,988	0	23.803
WSPT2 + NDR-D	36,241	31,259	0	32.414
WSPT3 + NDR-D	36,805	30,695	0	35.073
WPD1 + NDR-D	36,711	30,789	0	28.842
WPD2 + NDR-D	36,687	30,813	0	22.801
WPD3 + NDR-D	37,274	30,226	0	22.548
WPD4 + NDR-D	37,403	30,097	0	6.285
ATC1 + NDR-D	39,458	28,042	0	14.73
ATC2 + NDR-D	39,276	28,224	0	9.001
ATC3 + NDR-D	40,814	26,686	0	14.01
COV1 + NDR-D	49,173	18,327	0	39.027
COV2 + NDR-D	49,322	18,178	0	35.437
COV3 + NDR-D	48,147	19,353	0	40.799
COV4 + NDR-D	48,382	19,118	0	57.086
GA + NDR-D	37,238	30,262	0	22.006
SA + NDR-D	36,951	30,549	0	17.221

generated heuristics (EDD1, WSPT2, WPD1 and ATC1) are the best for TWT with double due date. The proposed NDR-D provides a sufficient condition for local optimality. SA and GAs work better than heuristics, and the best heuristic solutions are used in GAs and SA as the initial solution. The results show that the amount of improvement is significant. One of the methods to reach the performance criteria, expected of the shop-floor by management, is to examine the different priority rules. Too many priority rules are proposed for single due date. However, few are proposed for double due date. In this study, new priority rules are proposed and adapted according to double due date, and additionally, the results are improved by applying NDR-R to these determined priority rules. Furthermore, the best working priority rule for double due date can also be seen. The proposed method has practical applicability in industry from the viewpoint of the successful results. For future research, the single-machine TWT problem with double due date and unequal release date can be modeled by using artificial neural networks.

References

1. Hsu CJ, Yang SJ, Yang DL (2011) Two due date assignment problems with position dependent processing time on a single machine. Comput Ind Eng 60:796–800
2. Gordon VS, Strusevich VA (2009) Single machine scheduling and due date assignment with positionally dependent processing times. Eur J Oper Res 198(57–62):2009
3. Shabtay D, Steiner G (2006) Two due date assignment problems in scheduling a single machine. Oper Res Lett 34:683–691
4. Wang C (2011) Due-date management through iterative bidding. IEEE Trans Syst Man Cybern—Part A Syst Hum 41(6):1182–1198
5. Van den Akker JM, Diepen G, Hoogeveen JA (2010) Minimizing total weighted tardiness on a single machine with release dates and equal-length jobs. J Sched 13(6):561–576
6. Li JQ, Yuan XH, Lee ES, Xu DH (2011) Setting due dates to minimize the total weighted possibilistic mean value of the weighted earliness-tardiness costs on a single machine. Comput Math Appl 62(11):4126–4139
7. Kellegoz T, Toklu B, Wilson J (2008) Comparing efficiencies of genetic crossover operators for one machine total weighted tardiness problem. Appl Math Comput 199:590–598
8. Yoon SH, Lee IS (2011) New constructive heuristics for the total weighted tardiness problem. J Oper Res Soc 62(1):232–237
9. Adams J, Balas E, Zawack D (1988) The shifting bottleneck procedure for job shop scheduling. Manag Sci 34(3):391–401
10. Colka Altunc AB, Burak Keha A (2009) Interval-indexed formulation based heuristics for single machine total weighted tardiness problem. Comput Oper Res 36(6):2122–2131
11. Lawler EL (1997) A "Pseudopolynomial" algorithm for sequencing job to minimize total tardiness. Ann Discret Math 1:331–342
12. Chambers RJ, Carraway RL, Lowe TJ, Morin TL (1991) Dominance and decomposition heuristics for single machine scheduling. Oper Res 39:639–647
13. Emmons H (1969) One machine sequencing to minimize certain functions of job tardiness. Oper Res 17:701–715
14. Rinnooy Kan AHG, Lageweg BJ, Lenstra JK (1975) Minimizing total costs in one machine scheduling. Oper Res 23:908–927
15. Fisher ML (1976) A dual algorithm for the one-machine scheduling problem. Math Progr 11:229–251
16. Potts CN, Van Wassenhove LN (1985) A branch and bound algorithm for total weighted tardiness problem. Oper Res 33:363–377
17. Potts CN, Van Wassenhove LN (1987) Dynamic programming and decomposition approaches for the single machine total tardiness problem. Eur J Oper Res 32:405–414
18. Sabuncuoglu I, Gurgun B (1996) A neural network model for scheduling problems. Eur J Oper Res 93(2):288–299
19. Abdul-Razaq TS, Potts CN, Van Wassenhove LN (1990) A survey of algorithms for the single machine total weighted tardiness scheduling problem. Discret Appl Math 26:235–253
20. Bozejko W (2010) Parallel path relinking method for the single machine total weighted tardiness problem with sequence-dependent setups. J Intell Manuf 21(6):777–785
21. Cheng H, Yixun L, Ruyan F (2009) Bicriteria scheduling with double due dates to minimize the maximum lateness. Chin J Eng Math 26(1):147–150
22. Mahnam M, Moslehi G (2009) A branch-and-bound algorithm for minimizing the sum of maximum earliness and tardiness with unequal release times. Eng Optim 41(6):521–536
23. Li K, Yang SL, Ren ML (2011) Single machine scheduling problem with resource dependent release dates to minimize total resource-consumption. Int J Syst Sci 42(10):1811–1820

24. Geiger MJ (2010) On heuristic search for the single machine total weighted tardiness problem—some theoretical insights and their empirical verification. Eur J Oper Res 207:1235–1243
25. Eren T (2009) Minimizing the total weighted completion time on a single machine scheduling with release dates and a learning effect. Appl Math Comput 208:355–358
26. Kenneth KR, Keller B (2010) Solving the single-machine sequencing problem using integer programming. Comput Ind Eng 59:730–735
27. Vepsalainen APJ, Morton TE (1987) Priority rules for job shops with weighted tardiness cost. Manag Sci 33:1035–1047
28. Akturk MS, Yidirim MB (1998) A new lower bounding scheme for the total weighted tardiness problem. Comput Oper Res 25(4):265–278
29. Kanet JJ (2007) New precedence theorems for one-machine weighted tardiness. Math Oper Res 32(3):579–588
30. Jouglet A, Carlier J (2011) Dominance rules in combinatorial optimization problems. Eur J Oper Res 212:433–444
31. Akturk MS, Ozdemir D (2001) A new dominance rule to minimize total weighted tardiness with unequal release date. Eur J Oper Res 135:394–412
32. Cakar T (2005) A new neuro-dominance rule for single machine tardiness problem. Lect Notes Comput Sci 3483:1241–1250
33. Cakar T (2011) Single machine scheduling with unequal release date using neuro-dominance rule. J Intell Manuf 22:481–490
34. Chan FTS, Chan HK, Kazerooni A (2003) Real time fuzzy scheduling rules in FMS. J Intell Manuf 14(3–4):341–350
35. Dudek-Dyduch E (2000) Learning based algorithms in scheduling. J Intell Manuf 11(2):135–143
36. Laguna L, Barnes JW, Glover FW (1991) Tabu search methods for a single machine scheduling problem. J Intell Manuf 2(2):63–74
37. Weckman GR, Ganduri CV, Koonce DA (2008) A neural network job shop scheduler. J Intell Manuf 19(2):191–201
38. Yim SJ, Lee DY (1999) Scheduling cluster tools in wafer fabrication using candidate list and simulated annealing. J Intell Manuf 10(6):531–540
39. Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220(4598):671–680
40. Schlunz EB, Van Vuuren JH (2013) "An investigation into the effectiveness of simulated annealing as a solution approach for the generator maintenance scheduling problem. Electr Power Energy Syst 53:166–174
41. Köker R (2013) A neuro-simulated annealing approach to the inverse kinematics solution of redundant robotic manipulators. Eng Comput 29(4):507–515
42. Çakar T, Yazgan HR, Köker R (2008) Parallel robot manipulators, new developments. In: Ryu J-H (ed) Parallel robot scheduling with genetic algorithms. I-Tech Education and Publishing, Vienna, pp 153–170
43. Çakar T, Köker R, Demir I (2008) Parallel robot scheduling to minimize mean tardiness with precedence constraints using a genetic algorithm. Adv Eng Softw 39(1):47–54