International Journal of Engineering, Mathematical and Physical Sciences
ISSN: 2517-9934
Vol:8, No:6, 2014

Solving Single Machine Total Weighted Tardiness Problem Using Gaussian Process Regression

Wanatchapong Kongkaew

***Abstract*—**This paper proposes an application of probabilistic technique, namely Gaussian process regression, for estimating an optimal sequence of the single machine with total weighted tardiness (SMTWT) scheduling problem. In this work, the Gaussian process regression (GPR) model is utilized to predict an optimal sequence of the SMTWT problem, and its solution is improved by using an iterated local search based on simulated annealing scheme, called GPRISA algorithm. The results show that the proposed GPRISA method achieves a very good performance and a reasonable trade-off between solution quality and time consumption. Moreover, in the comparison of deviation from the best-known solution, the proposed mechanism noticeably outperforms the recently existing approaches.

***Keywords*—**Gaussian process regression, iterated local search, simulated annealing, single machine total weighted tardiness.

I. Introduction

THE single machine total weighted tardiness (SMTWT) problem is a special case of scheduling problem that is referred to be a strongly NP-hard problem [1]. This problem deals with the scheduling of a set of independent jobs $N_J = \{1,2,...,n_J\}$ to be processed without interruption on a single machine that can handle only one job at a time. Each job j has an integer processing time p_j, a due date d_j, and its priority established by weight w_j. For a given order of the jobs, the (earliness) completion time C_j and their tardiness $T_j = \max(C_j - d_j, 0)$ (i.e., the job is completed after its committed due date) are computed. The weighted tardiness $w_j T_j$ will occur when the job is not accomplished within its due date. Therefore, the total weighted tardiness for a given sequence π is computed as $Z(\pi) = \sum_{j \in N_j} w_j T_j$, and it is minimized to achieve the goal of this problem.

The SMTWT problem becomes considerable in real-world situations such as sequencing in production process, sequencing of aircrafts takeoff and landing, and assigning the sequence of stages in a construction project, delivering the goods with the customer's priority in supply chain, and so on [2]. Throughout the last decade, many researchers have proposed the approaches to solve the SMTWT problem. The well-known exact method is branch and bounds algorithms [3]–[5] to generate solutions that are guaranteed an optimality, but these algorithms encounter the obstacle on restriction of computational time or computer storage requirements, especially when the problem size exceeds 50 jobs [6].

Heuristics and meta-heuristics, in most published work, is attractive due to it can be fast solved to obtain near-optimal solution. Some of these schemes already used for the SMTWT problem include local search techniques [6], polynomial-time approximation algorithm [7], tabu search [8], [9], simulated annealing [10], ant colony optimization [11], iterated dynasearch [12], genetic algorithm [13], [14], GRASP with path relinking [15], variable neighborhood search [16], and variable structure learning automata [17]. Note that, the iterated dynasearch [12] is one of iterative methods that employs the dynamic programming combining with local search. In the contribution of this paper, we present a method that utilizes a probabilistic model (i.e., Gaussian process regression model) incorporated with an iterated local search to solve the SMTWT problem. In addition, the search performance of proposed algorithms is considered in both terms of solution deviation and computation time.

This paper is organized as follows. The conceptual framework of Gaussian process regression (GPR) and its applications are described in Section II. In Section III, the proposed GPR-based algorithm for solving the SMTWT problem has been presented. Next, Section IV provides the computational results and comparison with some recent methods. Finally, the overall result of this work is concluded in Section V.

II. Gaussian Process Regression

The Gaussian process regression (GPR) is known as a probabilistic approach for a regression model due to its practical and theoretical simplicity and excellent generalization ability [18]. The applications of GPR are found in many fields, for instance, the depth estimation of a point in the camera's image position [19], the optimization on the sensor placements in the art-gallery problem [20], the quality improvement in the wire-cut electrical discharge machining process [21], and the traffic problem in Japan [22]. In addition, it is applied to a well-known problem in operations research field at first with the traveling salesman problem [23]. Hence, the theoretical framework of GPR method is expressed as the following subsections.

A. GPR Model

Gaussian process (GP) is a collection of random variables, any finite number of which has a joint Gaussian distribution, and it is specified by its mean function $m(\mathbf{x})$ and covariance

W. Kongkaew is with the Department of Industrial Engineering, Faculty of Engineering, Prince of Songkla University, Songkhla, Thailand 90110 (phone: 667-428-7181; fax: 667-455-8829; e-mail: wanatchapong.k@psu.ac.th).

function $k(\mathbf{x},\mathbf{x}')$ [18]. Usually the mean function is assumed to be a zero function; thus, the Gaussian process can be written as $f(\mathbf{x}) \sim GP[0, k(\mathbf{x},\mathbf{x}')]$.

Gaussian process regression (GPR) is a model to estimate the value of a dependent variable or output by using some observations of dependent variables at certain values of the independent variable. Considering a training data set T_s of n observations, it is given as $T_s = \{(\mathbf{x}_r, \mathbf{y}_r) \mid r = 1,2,...,n\}$, where $\mathbf{x}_r$ denotes a column vector of the $r-$th sequence of n_J jobs and y_r denotes a scalar observation, i.e., the total weighted tardiness obtained from the $r-$th sequence. For convenience, let $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n]$ denote the inputs and let $\mathbf{y} = [y_1, y_2, ..., y_n]^{\mathrm{T}}$ be the observations, so the training data set can be written as $T_s = (\mathbf{X}, \mathbf{y})$.

The assumption of GPR is that the observation arises from some unknown function of $\mathbf{x}_r$, and it may be corrupted by unknown Gaussian noise ε_r. Thus, GPR model is given by

$$y_r = \phi(\mathbf{x}_r)^{\mathrm{T}} \mathbf{w} + \varepsilon_r,$$

where $\mathbf{w}$ is the vector of weight parameters, ε_r is a white Gaussian noise which follows an independent and identically Gaussian distributed with zero mean and noise variance σ_n^2, that is $\varepsilon_r \sim Normal(0, \sigma_n^2)$, where n is the number of observations.

Over all inputs and observations in a training dataset, the prior probability density of the observations given the parameters is estimated. Then, it is used to compute the posterior distribution over functions for making prediction (See the theoretical framework in [18], [23]). The prediction is done by the posterior mean and the posterior variance. Given a single test input $\mathbf{x}_s$, let $f_s \triangleq f(\mathbf{x}_s)$ be the function value at a single test input $\mathbf{x}_s$. The posterior mean (or the predictive function value) is given by

$$\hat{f}_s = \mathbf{k}_s^{\mathrm{T}} \mathbf{K}^{-1} \mathbf{y}, \tag{1}$$

where $\mathbf{K}$ is the covariance matrix evaluated at all possible pairs of training input, $\mathbf{k}_s^{\mathrm{T}}$ is the transpose of the vector of the covariance between training and test inputs. In addition, the (r,t) entry of the covariance matrix (or vector) is determined by the covariance function $k(\mathbf{x}_r, \mathbf{x}_t)$ [18]. Note that the predictive variance is not considered because we make the prediction at a single test input. Moreover, the graphical model for GPR can be shown in Fig. 1.

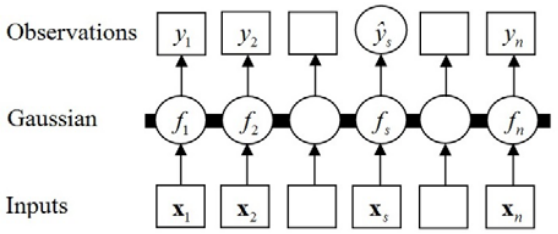

Fig. 1 Pictorial of GPR approach [18], [23]

B. Covariance Function

Many covariance functions can be used to define a GP prior, for example, Matérn class of covariance function, squared exponential (SE) covariance function, and radial basis covariance function [18]. However, the SE covariance function is applied in our proposed algorithm because it is the most popular kernel, and it is given by

$$k_{SE}(\mathbf{x}_r, \mathbf{x}_t) = \sigma_f^2 \exp\left[-\frac{1}{2\ell^2}(\mathbf{x}_r - \mathbf{x}_t)^{\mathrm{T}}(\mathbf{x}_r - \mathbf{x}_t)\right] + \sigma_n^2 \delta(\mathbf{x}_r, \mathbf{x}_t), \tag{2}$$

where $\delta(\mathbf{x}_r, \mathbf{x}_t)$ is the Kronecker delta function which equals to 1 if and only if $r = t$ and 0 otherwise, ℓ is the characteristic length-scale, σ_f^2 is the signal variance of function, and σ_n^2 is the white noise variance.

C. GPR Parameter Estimation

Given a kernel, the hyperparameters of the covariance function (i.e., the characteristic length-scale ℓ, the signal variance of function σ_f^2, and the white noise variance σ_n^2) are determined by the maximum likelihood method [18]. The log marginal likelihood function under the GP model is

$$\log p(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\theta}) = \frac{1}{2}\mathbf{y}^{\mathrm{T}}\boldsymbol{\alpha} - \frac{1}{2}\log\left|\mathbf{K} + \sigma_n^2\mathbf{I}\right| - \frac{n}{2}\log 2\pi, \tag{3}$$

where $\boldsymbol{\alpha} = \mathbf{K}^{-1}\mathbf{y}$ and $\boldsymbol{\theta}$ is a vector of ℓ, σ_f^2, and σ_n^2. The partial derivative of (3) with respect to $\boldsymbol{\theta}$ is minimized by using a gradient-based optimizer until converging to zero [18]. This result gives a vector of optimized hyperparameters.

In this section, we give a concise explanation on the theoretical framework of the GPR approach. The next section delineates step-by-step on its implementation, which incorporates with an iterated local search, for the SMTWT problem.

III. Proposed Algorithm for the SMTWT Problem

This section aims to discuss on our proposed approach for solving the single machine total weighted tardiness scheduling problem. This fashion combines the Gaussian process regression mechanism with the iterated local search technique (i.e., simulated annealing), then we call it as the "GPRISA" algorithm. The main procedure of proposed algorithm consists

of three phases: data preparation, prediction, and improvement phases, which are described respectively. In addition, the pseudocode of this method is disclosed at the end of this section.

A. Data Preparation Phase

This phase deals with the construction of sample sequences that provide for training input of GPR. In order to diversify the search space, four constructive rules and the double-bridge swapping strategy are used to construct the sample sequences. Initially, each constructive rule is employed to create the initial sequence of n_J jobs. These dispatching rules [13] are compendiously explained as follows:

1) Weighted shortest processing time (WSPT): Jobs are arranged to process on a machine in the ascending order of the ratio S_j, $S_j = p_j / w_j$.
2) Shortest processing time (SPT): Each job j is scheduled to process on a machine with the ascending order of its processing time.
3) Biggest weighted first (BWF): Each job j is scheduled to process on machine in the descending order of its weight.
4) The weighted modified due date (WMDD) is an effective rule for jobs sequencing with the total weighted tardiness, introduced by [24]. This dispatching rule outperforms other ones, e.g., EDD, WSPT, and WEDD (weighted EDD). The brief procedure is shown in Fig. 2:

Procedure *Weighted modified due date (WMDD)*

1. Set $t = 0$, $Seq = \varnothing$, and $Job = \{1, 2, 3, ..., n_J\}$
2. Compute the value of γ_j for each job in Job, as follow.
$$\gamma_j = \frac{\max\{p_j, d_j - t\}}{w_j}$$
3. Sort the value of γ_j in ascending order, and select job k having the minimum value of γ_j.
4. Add job k in Seq, then remove it from Job.
5. Update $t \leftarrow t + p_k$.
6. Terminate the procedure if Job is empty; otherwise, go to step 1.

Fig. 2 Pseudocode of the weighted modified due date rule

After constructing four initial sequences, an individual is perturbed again by the double-bridge scheme, which randomly selects four jobs in the given sequence and swaps those jobs [25]. This leads to new (additional) sequences as in demand. Here, we obtain various sample sequences (or observations) to be an input of GPR.

In order to treat the sample sequences as an input of GPR, they must be represented and they are available in convenient situations. Several mechanisms have been presented to represent a permutation of n_J jobs (or cities in the traveling salesman problem), for instance, path representation, binary string representation, binary matrix representation [26]. However, our proposed algorithm employs the binary string scheme to represent all sequences since it performs well when making prediction. This method encodes each job in a given sequence as a string of $\lceil \log_2(n_J) \rceil$ bits, and then a complete sequence becomes a string of $n_J \cdot \lceil \log_2(n_J) \rceil$ bits [26]; for example, a string of $[001 \quad 011 \quad 010 \quad 000]$ belongs to a sequence $2 \rightarrow 4 \rightarrow 3 \rightarrow 1$.

B. Prediction Phase

This phase concerns with the prediction of an optimal sequence of n_J jobs that process on the single machine. After preparing the training dataset in Section A, the sample sequences and their total weighted tardiness (TWT) values are employed to make prediction through GPR model, in which the sample sequences are treated as independent variables and their TWT values are dependent variables. In order to select a suitable model, the SE covariance function is computed over all possible pairs of two sequences by using (2), and then it is minimized the log marginal likelihood function (as defined in (3)) to determine the hyper parameters of the kernel that are consistent for the given training inputs.

In the prediction step, the optimized hyper parameters are utilized to compute posterior distribution over functions. At this step, a single test input is given and used to compute the posterior mean (or the predictive function value) by using (1). Note that the single test input in this step is a sequence that has a minimum TWT value in training input. Later, the predictive function value is employed to indicate the index value r, given by

$$r = \arg\min_r \left[\left(y_r - \hat{f}_s \right)^2 \right].$$

Consequently, the sequence (in training input) that corresponds to r th index will be deputized an optimal sequence, and it provides for an input of the next phase. Moreover, in this step, we modify the GPR codes from their original in the GPML toolbox [27].

C. Improvement Phase

Several local searches have been proposed to improve an obtained sequence, e.g., descent methods, simulated annealing, threshold accepting, genetic algorithm, and tabu search [6], [13]. However, our proposed algorithm implements the iterated local search based on the simulated annealing (SA) algorithm (called the iterated SA or the ISA algorithm) because it is one of the provably optimal local searches [25], [28]–[30]. The ISA technique applies SA as a local search to some initially given sequence at the beginning of the algorithm, and then a main loop is iterated until some stopping criterion is satisfied.

In each main loop of the ISA mechanism, the modification step ("kick-move") yields a new locally optimal solution according to the SA local search and a previous solution, then these two solutions are evaluated in the acceptance step before starting the next loop. In addition, the SA scheme starts from

an initially given sequence at a high temperature that is gradually decreased. A new sequence is generated and the difference in the TWT values between a current sequence and a new candidate sequence is calculated; then the new sequence is accepted to be a current sequence if it is better, or it is accepted with some probability [29], [30].

In some practical aspects of ISA searching, the solution may be stuck at some locally optimal solutions for a long time; consequently, the ISA algorithm should restart with a new solution under some criterion. We apply the double-bridge method (which is explained in the data preparation phase) for restarting the solution of the ISA algorithm to escape from current local optima.

In order to clarify, Fig. 3 shows the procedure of the proposed GPRISA algorithm, which applies for solving a single machine scheduling with total weighted tardiness problem.

Procedure GPRISA algorithm

Notation:

$\hat{\mathbf{x}}^*$ (Predictive optimal-sequence), $y(\hat{\mathbf{x}}^*)$ (Total weighted tardiness of $\hat{\mathbf{x}}^*$), T_k (Initial temperature), β (Cooling rate), $u \in \mathbb{Z}^+$ (Inner-loop number), U (Maximum number of u), $k \in \mathbb{Z}^+$ (Outer-loop number), K (Maximum number of k), MaxCount_u (Stopping criterion of the unchanged TWT after u iterations), MaxCount_k (Restarting criterion of the immutable TWT after k iterations), MaxCount_*restart* (Stopping criterion after employing multi-restart points technique), *Seq_Obs* (the number of generated sequences), n_J is the number of jobs in a sequence.

Initialize: $\hat{\mathbf{x}}^*_{\text{best}} \leftarrow \hat{\mathbf{x}}^*$, $y(\hat{\mathbf{x}}^*_{\text{best}}) \leftarrow y(\hat{\mathbf{x}}^*)$, Count_*restart* $= 0$, Count_$u = 0$, Count_$k = 0$.

Main procedure:

%% *Dataset Preparation phase* %%

Construct the initially four sequences of n_J jobs by using dispatching rules: WSPT, SPT, BWF, and WMDD.
Apply the double bridge method to each sequence for generating other different sequences as *Seq_Obs*.
Encode all sequences as binary and aggregate them into a matrix $\mathbf{X}$.
Calculate the total weighted tardiness (TWT) of all sequences and aggregate into a vector $\mathbf{y}$.
Identify a sequence correspond to a minimum TWT in $\mathbf{X}$, and set it as a test input $\mathbf{x}_s$.

%% *GPR prediction phase* %%

Compute the SE covariance function of all possible pairs $(\mathbf{x}_r, \mathbf{x}_t)$.
Minimize the log marginal likelihood function, and then make the prediction to obtain the predictive value $\hat{f}_s$.
Find the $r-$th binary vector in $\mathbf{X}$ that corresponds to the minimum value of squared difference between each member of $\mathbf{y}$ and $\hat{f}_s$, given by

$$r = \arg\min_r \left[\left(y_r - \hat{f}_s \right)^2 \right].$$

Decode the obtained binary vector to a sequence in the jobs' (integer) number.

%% *Iterated simulated annealing phase* %%

for $k = 1$ to K do

%% *Application of simulated annealing scheme* %%

for $u = 1$ to U do
Generate a neighbor $\hat{\mathbf{x}}^{**}$ by random 2-exchange swapping of $\hat{\mathbf{x}}^*$.
Calculate the change in the TWT, $\Delta_{\hat{\mathbf{x}}^*,\hat{\mathbf{x}}^{**}}$.
Accept the solution $\hat{\mathbf{x}}^{**}$ if $\Delta_{\hat{\mathbf{x}}^*,\hat{\mathbf{x}}^{**}} \le 0$, then update a new best solution $y(\hat{\mathbf{x}}^{**}) \le y(\hat{\mathbf{x}}^*_{\text{best}})$,
else accept with probability: $RN_k < \exp\left(-\Delta_{\hat{\mathbf{x}}^*,\hat{\mathbf{x}}^{**}} / T_k\right)$.
Count the number of unimproved iteration u if the best TWT is unaltered.
If the number of count reaches MaxCount_u, then end the exploration in loop u.
End for

Update the temperature: $T_k \leftarrow \beta T_k$.
Update the new best solution of iteration k.
Count the number of unimproved iteration k if the best solution is unaltered.
If the number of count reaches MaxCount_k and the number of restart is available (Count_*restart* $\neq$ MaxCount_*restart*), then perturb the solution with the double bridge technique, else terminate the search in loop k.
End for

Return: the best sequence $\hat{\mathbf{x}}^*_{\text{best}}$, and its TWT value (i.e., $y(\hat{\mathbf{x}}^*_{\text{best}}) \triangleq \hat{y}^*_{\text{best}}$).

Fig. 3 Pseudocode of the GPRISA algorithm

IV. Computational Experiments and Results

This section describes the experimental conditions that we utilize to achieve the numerical results. In addition, we provide the comparison of the obtained results from our proposed method with the best-known (or optimal) solutions taken from OR library [31]. Moreover, the results obtained from some previous approaches are utilized to compare with our proposed algorithm only on the common instances.

A. Experimental Settings

In this work, our proposed algorithm is implemented in MATLAB program, and it is executed on three problem sets (with 40, 50, and 100 jobs of the problem size), in which each one contains 40 instances (i.e., 120 of total instances), taken from the OR library [31]. In addition, the computation is done on PC running Intel(R) Core(TM) i3-2100 CPU3.10 GHz. processor with 4 GB of memory. For the GPR prediction, the training dataset obtains from 20 sample sequences and initial hyper parameters: $\ell = 2$, $\sigma_f^2 = 1$, and $\sigma_n^2 = 0$ (by trial and error).Moreover, the parameters used in our algorithm (in Fig. 3) are: $Seq\_Obs = 4$, $T_k = 1000$, $\beta = 0.97$, $U = 20000$, $K = 20000$, $\text{MaxCount}\_k = 50$, $\text{MaxCount}\_u = 50$, $\text{MaxCount}\_restart = 20$ for $n_J \le 50$ jobs, and $\text{MaxCount}\_restart = 30$ for $n_J = 100$ jobs.

Because our algorithm needs a training dataset that we create randomly, we repeat the GPRISA algorithm on each SMTWT instance for five times. Then, the total weighted tardiness values are averaged. Furthermore, the performance of our proposed algorithm measures the deviation percentage of its solution (i.e., an average of $\hat{y}^*_{\text{best}}$ values achieved by our GPRISA algorithm) from the best-known (or optimal) solution (y^*) correspond to instance, given by

$$\%\ \text{of deviation} = \frac{\hat{y}^*_{\text{best}} - y^*}{y^*} \times 100, \qquad (4)$$

and the computational time is evaluated over the problem size.

B. Computational Results

The results of the proposed algorithm correspond to the average performance of five trials: the mean percentage of the average deviation of its solution from the best-known solutions for each problem set and the average running time are shown in Table I. For an individual instance in each problem set, the percentage of the average deviation among five trials is calculated by using (4), and then for all 40 instances the mean percentage of average deviation (MAD) is computed again.

Table I illustrates that the MAD values for the wt40, wt50, and wt100 problem sets (which each of them contains 40 instances) are 0.0109%, 0.0231%, and 0.0570%, respectively. These values indicate that the obtained solutions achieved by the GPRISA algorithm is extremely close to the best-known solutions (or optimal) solutions, which are provided in the OR library. However, the effort of computation is spent much time for searching, and it has an increasing time to achieve the near-optimal solution when the size of dataset becomes large. In our preliminary investigation, the GPRISA algorithm quickly reaches the best-known solution within short-term iterations, but it still executes until it meets the stopping criteria. Consequently, it takes more computational time than necessary to achieve the solution.

TABLE I
Computational Results for GPRISA Algorithm

Problem set	MAD (%)	Mean running time (seconds)
wt40	0.0109	31.3
wt50	0.0231	36.7
wt100	0.0570	67.9

For the precision and accuracy of searching, the median, first quartile, and third quartile of percentage of the average deviation occur with the same valueat zero while there is non-zero percentage of the average deviation for some instances in the wt40 problem (as in Fig. 4). For the wt50 problem, there are only the median and first quartile of one that is absorbed at zero. Lastly, for the wt100 problem, the distribution of percentage of the average deviation gets slightly away from zero; however, the minimum percentage of that is still at zero.

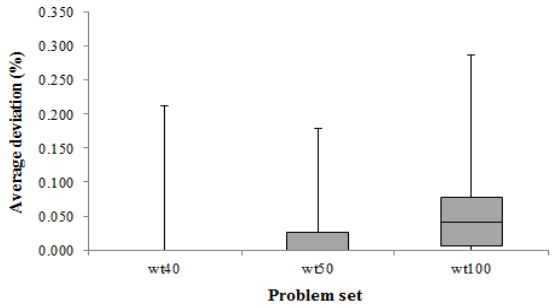

Fig. 4 Percentage of the average deviation obtained from each problem set

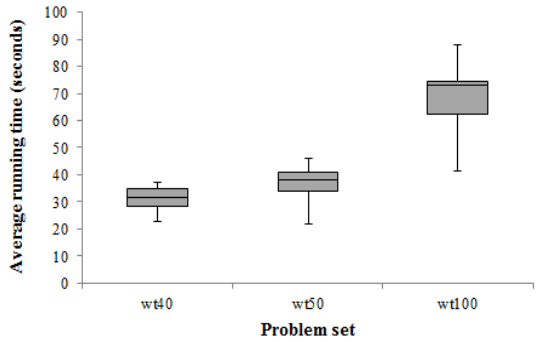

Fig. 5 The average running time executed on each problem set

Fig. 5 shows the box-and-whisker plot of average running time among three datasets (consisting of 40 instances in each dataset) consumed by the GPRISA algorithm. This algorithm

spends an increasing time as the exponential trend (depending on the problem size). For the wt40 and wt50 problems, the GPRISA algorithm spends approximately 31 seconds and 35 seconds of the executing time respectively while it consumes roughly 70 seconds of the running time on the wt100 dataset.

C. Comparison

This section discusses the comparison of our proposed GPRISA algorithm with two recent approaches: the backward forward (BF) heuristics [2] and the variable structure learning automata (VSLA) [17], on the common 26 instances (belonging to three problem sets). The summary results for comparison, including the common instance number in each problem size, the best-known solution for each instance, the mean percentages of average deviation obtained by GPRISA, the percentages of average deviation achieved by BF heuristics, and the amount of deviation percentages produced by VSLA algorithm, can be seen in Table II.

TABLE II
COMPARISON OF THE SEARCH PERFORMANCE

Problem set	Instance number	Best-known solutions	Methods (unit: percent)		
			BF heuristics [2]	VSLA [17]	GPRISA
wt40 (Problem size = 40 jobs)	1	913	0.0000	13.075	0.0000
	9	16225	2.7304	2.648	0.0000
	11	17465	0.5554	2.961	0.0000
	24	119947	0.4911	0.211	0.0000
	59	3784	0.0000	17.381	0.0000
	89	25881	3.3306	5.227	0.0240
	97	114686	0.3688	0.337	0.0000
	119	66707	0.6461	1.665	0.0000
wt50 (Problem size = 50 jobs)	12	36378	0.4508	4.019	0.1792
	59	3770	4.1379	16.769	0.1326
	113	35106	4.7884	6.295	0.0000
	120	101665	0.6433	1.189	0.0504
	121	78315	0.6857	0.709	0.0743
wt100 (Problem size = 100 jobs)	14	157476	5.8060	9.401	0.0563
	24	744287	2.7632	0.154	0.0005
	31	24202	4.5327	25.276	0.1289
	38	90440	9.3941	9.306	0.0621
	42	425875	4.5537	0.797	0.0524
	47	623356	4.2560	0.186	0.0028
	60	19912	7.5331	38.129	0.0000
	71	640816	2.8319	0.165	0.0050
	89	54612	7.8096	6.427	0.0897
	99	622464	4.8980	0.175	0.0058
	111	159123	5.1243	2.431	0.0220
	116	370614	7.3513	0.967	0.0403
	123	397029	5.8991	0.379	0.0345

In Table II, for all the 26 common instances, our proposed GPRISA algorithm performs remarkably better amount of deviation percentages than both comparing methods. For the wt40 problem, our GPRISA algorithm achieves definitely the best-known solutions that belong to almost all instances, except for the instance number 89 with 0.024% of deviation from its best-known value). In addition, for the wt50 and wt100 problems, there is only an instance of each problem that the GPRISA innovation accomplishes the best-known solution, i.e., the instance number 113 of the wt50 problem and the instance number 60 of the wt100 problem.

In summary, our proposed GPRISA algorithm is the best method, among three mechanisms. It admirably reaches the best-known solutions of nine instances (out of 26). Finally, almost all amounts of deviation percentages over 26 common instances are lower than 0.1%, from the best-known solutions.

V. CONCLUSION

This paper develops a new algorithm, which combines a probabilistic method (namely, Gaussian process regression) with an iterated local search based on simulated annealing scheme, for solving the single machine total weighted tardiness-scheduling problem. It adopts many construction heuristics: WSPT, SPT, BWF, and WMDD, to construct the sample sequences, and then each of individuals are employed to generate other sample sequences by using the double-bridge method. These sequences and their sums of weighted tardiness are treated as a training input of the Gaussian process regression (GPR) for making a prediction of an optimal sequence. After that, the iterated simulated annealing (ISA) technique is called for improving the obtained solution. Hence, we call this method: GPRISA algorithm.

In order to investigate on the search performance, we provide the numerical experiments on various problem sizes

with the 120 benchmark instances (40 instances for each problem size of 40, 50, and 100 jobs) taken from OR library. As a result, the proposed GPRISA algorithm performs an efficiency performance. In addition, for all three problem sizes, it achieves very close to the best-known (or optimal) solutions within 0.1% of average deviation percentages, and the computational time spent to achieve solution is reasonable. Moreover, we compare the search performance of the GPRISA algorithm with two existing methods: the backward forward (BF) heuristics and the variable structure learning automata (VSLA), on the common 26 instances (belonging to three problem sets). The results show that our proposed GPRISA algorithm clearly outperforms the comparing approaches for all 26 common instances.

The future work aims to improve the performance of our proposed GPRISA algorithm to reduce the computational time that spends to acquire the optimal values. Furthermore, the application of the proposed algorithm will be considered for implementing to other optimization problems.

ACKNOWLEDGMENT

The author gratefully acknowledges the financial support from Prince of Songkla University in carrying out this research.

REFERENCES

[1] J.K. Lenstra, A.H.G. RinnooyKan, and P. Brucker,"Complexity of machine scheduling problems,"in*Annals of Discrete Mathematics*, P. L. Hammer, E. L. Johnson, B. H. Korte, and G. L. Nemhauser, Eds. Amsterdam: Elsevier, 1977, pp. 343–362.

[2] R. Maheswaran and S. G. Ponnambalam, "An investigation on single machine total weighted tardiness scheduling problems," *Int. J. Adv. Manuf. Tech.*, vol. 22, pp. 243–248, September 2003.

[3] A. Jouglet, P. Baptiste, and J. Carlier, "Exact procedures for single machine total cost scheduling," in *Proc. 2002 IEEE Int. Conf. Systems, Man and Cybernetics*, Tunisia, 2002, 4 p.

[4] P. Babu, L. Peridy, and E. Pinson, "A branch and bound algorithm to minimize total weighted tardiness on a single processor," *Ann. Oper. Res.*, vol. 129, pp. 33–46, July 2004.

[5] M. Wodecki, "A branch-and-bound parallel algorithm for single-machine total weighted tardiness problem," *Int. J. Adv. Manuf. Tech.*, vol. 37, pp. 996–1004, June 2008.

[6] H. A. J. Crauwels, C. N. Potts, and L. N. Van Wassenhove, "Local search heuristics for single machine total weighted tardiness scheduling problem," *Informs J. Comput.,* vol. 10, pp. 341–350, Summer 1998.

[7] T. C. E. Cheng, C. T. Ng, J. J. Yuan, and Z. H. Liu, "Single machine scheduling to minimize total weighted tardiness," *Eur. J. Oper. Res.*, vol. 165, pp. 423–443, September 2005.

[8] W. Bożejko, J. Grabowski, and M. Wodecki, "Block approach—tabu search algorithm for single machine total weighted tardiness problem," *Comput. Ind. Eng.*, vol. 50, pp. 1–14, May 2006.

[9] Ü. Bilge, M. Kurtulan, and F. Kıraç, "A tabu search algorithm for the single machine total weighted tardiness problem," *Eur. J. Oper. Res.*, vol. 176, pp. 1423–1435, February 2007.

[10] A. C. Nearchou, "Solving the single machine total weighted tardiness scheduling problem using a hybrid simulated annealing algorithm," in *Proc. 2nd IEEE Int. Conf. Industrial Informatics*, New Jersey, 2004, pp. 513–516.

[11] O. Holthaus and C. Rajendran, "A fast ant-colony algorithm for single-machine scheduling to minimize the sum of weighted tardiness of jobs," *J. Oper. Res. Soc.*, vol.56, pp. 947–953, August 2005.

[12] R. K. Congram, C. N. Potts, and S. L. Van de Velde, "An iterated dynasearch algorithm for the single-machine total weighted tardiness scheduling problem," *Informs J. Comput.*, vol. 14, 52–67, Winter 2002.

[13] N. Liu, M. Abdelrahman, and S. Ramaswamy, "A genetic algorithm for single machine total weighted tardiness scheduling problem," *Int. J. Intelligent Control and Systems*, vol. 10, pp. 218–225, September 2005.

[14] A. Ferrolho and M. Crisostomo, "Single machine total weighted tardiness problem with genetic algorithms," in *Proc. 2007 IEEE/ACS Int. Conf. Computer Systems and Applications*, Amman, Jordan, 2007, pp. 1–8.

[15] J. E. C. Arroyo, A. G. Santos, F. L. S. Silva, and A. F. Araújo, "A GRASP with path relinking for the single machine total weighted tardiness problem," in *Proc. 8th Int. Conf. Hybrid Intelligent Systems*, Barcelona, Spain, 2008, pp. 726–731.

[16] X. Wang and L. Tang, "A population-based variable neighborhood search for the single machine total weighted tardiness problem," *Comput.Oper. Res.*, vol. 36, pp. 2105–2110, June 2009.

[17] S. Sabamoniri, K. Asghari, and M. J. Hosseini, "Solving single machine total weighted tardiness problem using variable structure learning automata," *Int. J. Comput. Appl.*, vol. 56, pp. 37–42, October 2012.

[18] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*, Cambridge: MIT Press, 2006, ch. 2.

[19] F. H. Sinz, J. Q. Candela, G. H. Bakir, C. E. Rasmussen, and M. O. Franz, "Learning depth from stereo," in *Proc. 26th DAGM Symp. Pattern Recognition*, Tübingen, Germany, 2004, pp. 245–252.

[20] A. Krause, A. Singh, and C. Guestrin, "Near-optimal sensor placements in Gaussian processes: theory, efficient algorithms and empirical studies," *J. Mach. Learn. Res.*, vol. 9, pp. 235–284, February 2008.

[21] J. Yuan, K. Wang, T. Yu, and M. Fang, "Reliable multi-objective optimization of high-speed WEDM process based on Gaussian process regression," *Int. J. Mach. Tool.Manu.*, vol. 48, pp. 47–60, January 2008.

[22] T. Idé and S. Kato, "Travel-time prediction using Gaussian process regression: a trajectory-based approach," in *Proc. 9th SIAM Int. Conf. Data Mining*, Nevada, USA, 2009, pp. 1185–1196.

[23] W. Kongkaew and J. Pichitlamken, "A Gaussian process regression model for the traveling salesman problem," *J. Comput. Sci.*, vol. 8, pp. 1749–1758, August 2012.

[24] J. J. Kanet and X. Li, "A weighted modified due date rule for sequencing to minimize weighted tardiness," *J. Sched.*, vol. 7, pp. 261–276, July 2004.

[25] H.R. Lourenço, O.C. Martin, and T. Stützle, "Iterated local search: framework and applications," in *Handbook of Metaheuristics*, M. Gendreau and J.Y. Potvin, Eds. New York: Springer-Verlag, 2010, pp. 363–397.

[26] P. Larrañaga, C. M. H. Kuijpers, R. H. Murga, I. Inza, and S. Dizdarevic, "Genetic algorithms for the travelling salesman problem: a review of representations and operators," *Artif. Intell. Rev.*, vol. 13, pp. 129–170, April 1999.

[27] C. E. Rasmussen and H. Nickisch, "Gaussian processes for machine learning (GPML) toolbox," *J. Mach. Learn. Res.*, vol. 11, pp. 3011–3015, November 2010.

[28] A.W. Johnson and S.H. Jacobson, "A class of convergent generalized hill climbing algorithms," *Appl. Math.Comput.*, vol. 125, pp. 359–373, January 2002.

[29] E.H.L. Aarts, J.H.M. Korst, and P.J.M. Van Laarhoven, "Simulated annealing," in *Local Search in Combinatorial Optimization*, E. Aarts and J.K. Lenstra, Eds. Princeton: Princeton University Press, 2003, pp. 91–120.

[30] E. Aarts, J. Korst, and W. Michiels, "Simulated annealing," in *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, E.K. Burke and G. Kendall, Eds. New York: Springer-Verlag, 2005, pp. 187–210.

[31] J.E.Beasley,"OR-Library,"Availablesource: http://people.brunel.ac.uk/~mastjjb/jeb/orlib/wtinfo.html, June 2012.

WanatchapongKongkaew is currently the lecturer in the Department of Industrial Engineering at Prince of Songkla University, Thailand. He earned a D.Eng. in Industrial Engineering from Kasetsart University, Thailand in 2012 and a M.Eng. in Industrial and Systems Engineering in 2007 from Prince of Songkla University, Thailand. He also earned a B.Eng. in Industrial Engineering from Kasetsart University in 2004. His main research interests include applied operations research and statistics, stochastic modeling and simulation, logistics and supply chain engineering, and meta-heuristics.