Knowledge-Based Systems 90 (2015) 70–80

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

A knowledge-based evolutionary proactive scheduling approach in the presence of machine breakdown and deterioration effect

Du-Juan Wang[a], Feng Liu[b], Yan-Zhang Wang[a], Yaochu Jin[a,c,*]

[a] School of Management Science and Engineering, Dalian University of Technology, Dalian 116023, PR China
[b] School of Management Science and Engineering, Dongbei University of Finance and Economics, Dalian 116025, PR China
[c] Department of Computer Science, University of Surrey, Guildford, Surrey GU2 7XH, United Kingdom

ARTICLE INFO

Article history:
Received 14 May 2015
Revised 29 August 2015
Accepted 30 September 2015
Available online 20 October 2015

Keywords:
Proactive scheduling
Stochastic machine breakdown
Support vector regression
A priori domain knowledge
Robustness
Evolutionary algorithms

ABSTRACT

This paper considers proactive scheduling in response to stochastic machine breakdown under deteriorating production environments, where the actual processing time of a job gets longer along with machine's usage and age. It is assumed that a job's processing time is controllable by allocating extra resources and the machine breakdown can be described using a given probability distribution. If a machine breaks down, it needs to be repaired and is no longer available during the repair. To absorb the repair duration, the subsequent unfinished jobs are compressed as much as possible to match up the baseline schedule. This work aims to find the optimal baseline sequence and the resource allocation strategy to minimize the operational cost consisting of the total completion time cost and the resource consumption cost of the baseline schedule, and the rescheduling cost consisting of the match-up time cost and additional resource cost. To this end, an efficient multi-objective evolutionary algorithm based on elitist non-dominated sorting is proposed, in which a support vector regression (SVR) surrogate model is built to replace the time-consuming simulations in evaluating the rescheduling cost, which represents the solution robustness of the baseline schedule. In addition, a priori domain knowledge is embedded in population initialization and offspring generation to further enhance the performance of the algorithm. Comparative results and statistical analysis show that the proposed algorithm is effective in finding non-dominated tradeoff solutions between operational cost and robustness in the presence of machine breakdown and deterioration effect.

1. Introduction

As one of the most crucial functions in a manufacturing system, production/machine scheduling determines the allocation of limited resources, such as machines, operators and tools, to a set of competing jobs or operations on a short term (daily or weekly) basis, in order to optimize one or several objectives with respect to a job's completion time [1]. Production scheduling returns a baseline schedule which specifies the time/machine/operation assignments. The main purpose is to pursue the optimality or near-optimality of the baseline schedule under ideal environmental conditions. However, in practice, the assumption of ideal environmental condition does not hold due to the intrinsic uncertainties in real world. In the presence of such uncertainties, the baseline schedule quickly becomes infeasible as jobs scheduled in the time interval of machine breakdown could not be processed as planned, and therefore, appropriate reactions are needed to partially or completely reschedule the unfinished jobs. From the practitioner's point of view, it is important to make sure that the revised schedule deviates as little as possible from the baseline schedule and maintains satisfactory performance.

To handle such uncertainties, the robustness of the baseline schedule has been widely studied in the literature [2]. Generally speaking, there are two types of robustness: quality robustness and solution robustness. Quality robustness means that the performance of the realized schedule is relatively insensitive to machine breakdown, and does not degrade significantly in the presence of uncertainties. By contrast, solution robustness is also referred to as stability [3,4], which means that the realized schedule stays in consistence with the baseline schedule as much as possible after disruptions. The importance of solution robustness can be illustrated in three aspects [5]. First, facilitated by powerful Internet technologies, companies frequently share their production schedules with their raw material suppliers. It is expected that suppliers make just-in-time delivery of material following the baseline schedule. Second, the baseline schedule serves as a performance indicator for management and shop-floor operators. Third, the baseline schedule provides visibility into the near future, allowing the quotation of competitive delivery dates for customers.

* Corresponding author. Tel.: +44 1483686037.
E-mail address: yaochu.jin@surrey.ac.uk (Y. Jin).

http://dx.doi.org/10.1016/j.knosys.2015.09.032
0950-7051/© 2015 Elsevier B.V. All rights reserved.

Once machine breakdown occurs, repairing of the machine will be undertaken immediately, resulting in an unavailability time interval during which no production can be carried out. A widely adopted approach to reducing the impact of machine breakdown, known as the proactive scheduling, is to generate a predictive schedule that is robust against anticipated disruptions that may occur during execution of the schedule [2–4]. After disruption happens, the realized schedule can match up with the baseline schedule as soon as possible using additional resource cost. Therefore, this paper adopts the proactive scheduling approach and aims to find the optimal processing sequence and a resource allocation strategy so as to minimize the operational cost of the baseline schedule and the rescheduling cost in response to machine breakdown, which is in essence the solution robustness. Here, we assume that machine breakdown can be described using a probability distribution. It is further assumed that the operational cost is measured by the sum of total completion time cost and resource cost of the baseline schedule, and the rescheduling cost consists of the match-up time cost and additional resource cost. To minimize the above objectives, a multi-objective evolutionary algorithm based on non-dominated sorting has been proposed. To enhance the computational efficiency, support vector regression based surrogate models have been employed to reduce the extra computation time needed for assessing the solution robustness. In addition, a priori domain knowledge, here the structural property of the predictive schedule is embedded into the evolutionary algorithm to help improve the search efficiency.

The remainder of this paper is organized as follows. In Section 2, a review of relevant literature is provided. Section 3 formulates the proactive scheduling problem considered in this work. A knowledge-based multi-objective evolutionary algorithm is presented in Section 4 to solve the proposed proactive scheduling problem. In Section 5, comparative studies are conducted to verify the effectiveness of the proposed algorithm. Finally Section 6 concludes this paper and suggests a few future research directions.

2. Literature review

In scheduling literature, proactive scheduling approaches to handling uncertainties aim to prepare a baseline schedule which can be easily adjusted within little performance degradation [2–4]. Aytug et al. [6] review existing literature on scheduling in the presence of unforeseen disruptions and robust scheduling approaches focusing on predictive schedules that minimize the effect of disruptions. Sabuncuoglu and Goren [7] summarize existing robustness and stability measures for proactive scheduling and endeavor to understand the philosophy of proactive and reactive approaches by analyzing the major issues in a scheduling process under uncertainties and studying how different policies are generated for handling these issues.

The buffering approach is the most frequently used proactive approach to minimizing the impact of stochastic disruptions, through which idle times are inserted into the predictive schedules [8–10]. It is first proposed by Mehta and Uzsoy [8] for a job-shop scheduling problem. They present a proactive scheduling approach so as to absorb the impacts of breakdowns. By jointly deciding the location and amount of additional idle time along with the proactive schedule, a revised schedule can be obtained with simple adjustments and little performance degradation under certain conditions. After that, studies on accommodating disruptions through inserting idle times into the baseline schedule have been reported by many researchers. Leus and Herroelen [9] consider inserting idle times on a single machine to maximize solution stability under uncertainties. They show that the horizon of the predictive schedule is not propagated with small changes caused by uncertainty in processing time. Yang and Geunes [10] consider a scheduling problem aiming to create a predictive schedule with uncertain arrival of future jobs. The amount and positions of additional idle time are determined for the predictive schedule to minimize the sum of tardiness cost, disruption cost and wasted idle time cost. Goren and Sabuncuoglu [4] consider scheduling problems with processing time variability and machine breakdowns. A proactive scheduling approach that requires inserting idle times is taken so as to minimize two robustness and three stability measures being defined.

In the above reviewed literature, the buffering approach has been widely adopted [2,8,11–13], and has been examined as an efficient proactive approach to generating a robust baseline schedule with fixed processing times. However, inserting idle times will degrade the performance of the baseline schedule and if no disruption occurs, the inserted idle time becomes useless and the limited capacity of production resources is wasted. By assuming that the job processing time can be compressed with certain extra resource cost [14], several researchers consider absorbing disruptions by extensively compressing a set of jobs in the schedule to catch up with the baseline schedule at a certain point. For example, Akturk et al. [15] study a scheduling problem on non-identical parallel machines with disruptions, where the processing time of each job is controllable at a certain manufacturing cost. They generate reactive schedules to catch up with the baseline schedule as soon as possible in response to disruptions with a slight increase in manufacturing cost. Gurel et al. [16] consider anticipative scheduling problems on non-identical parallel machines with disruptions and controllable processing times. Distributions of uncertain events and flexibilities of jobs are considered for making the anticipative schedule, and a match-up strategy is used to catch up with the predictive schedule at some certain point with compression of processing times for the remaining jobs. Al-Hinai and ElMekkawy [17] address flexible job shop scheduling problems with random machine breakdowns to find robust and stable solutions. Several bi-objective measures along with the robustness and stability of the predictive schedule are investigated.

Deterioration effect widely exists in realistic production environments [18–23], and has been extensively studied in rescheduling problem with disruptions [24–26], where the actual processing time of a job becomes longer if the job starts processing later. However, not much work has considered generating a proactive schedule without idle time in response to machine breakdown under deteriorating production environments. Controllability of job's processing time enables us to use non-buffering proactive approaches so that the disruption can be absorbed at the price of additional resource cost. Taking deterioration effect into account makes the formulation of the problem more realistic but unfortunately also more difficult to solve.

Consequently, this work aims to solve proactive scheduling problems in the presence of machine breakdown and deterioration effect. The objective is to find an optimal processing sequence and a resource allocation strategy for the baseline schedule to simultaneously minimize the initial operational cost and rescheduling cost (i.e., solution robustness) in response to machine breakdown. In order to solve this challenging proactive scheduling problem, a knowledge-based multi-objective evolutionary algorithm is proposed to reduce the computational cost resulting from simulation-based robustness evaluation and to enhance the search capability of the algorithm.

3. Problem formulation

Assume that there is a set of n jobs $\{J_1, J_2, \ldots, J_n\}$ to be processed without interruption on a common machine. All jobs are available for processing at time zero. Each job J_j has a normal processing time $\bar{p}_j$. The processing times of jobs may be subject to change due to deterioration of the machine's performance with increase of machine's usage and age. In other words, the actual processing time of a job becomes longer if the job starts processing later. The strategy of controlling the processing times of jobs by allocating extra resource with corresponding resource consumption cost is adopted to improve the scheduling performance, assuming that the actual processing time of a job can be modeled as a linear function of the resource amount allocated to it. Specifically, the actual processing time of a job J_j is defined as: $p_j = \bar{p}_j + \alpha t_j - b_j u_j$, $0 \le u_j \le \bar{u}_j < \bar{p}_j / b_j$, where $\alpha > 0$ is the deteriorating rate shared by all jobs, t_j is the starting time of job J_j, b_j is the positive compression rate of job J_j, $\bar{u}_j$ is the maximum available resource amount and u_j is the amount of resource allocated to job J_j.

It is assumed that the decision maker is interested in finding a baseline schedule π to minimize the operational cost measured by the sum of the total completion time cost and resource consumption cost, which is one of the objectives under consideration, and is denoted as follows:

$$f(\pi) = q \sum_{j=1}^{n} C_j(\pi) + \sum_{j=1}^{n} c_j u_j \tag{1}$$

where q is the unit completion time cost, $C_j(\pi)$ is J_j's completion time in π, and c_j denotes the unit resource cost of job J_j. Using the three-field notation representation scheme $\boldsymbol{\alpha}|\boldsymbol{\beta}|\boldsymbol{\gamma}$ introduced by Graham et al. [27], we can formally formulate our original operational cost minimization problem denoted by $\boldsymbol{P}$ as follows:

$$1 \left| \bar{p}_j + \alpha t_j - b_j u_j \right| f(\pi) \tag{2}$$

where '1' stands for a single machine, '$\bar{p}_j + \alpha t_j - b_j u_j$' denotes job's actual processing time model, and $f(\pi)$ defines the objective to be optimized. $f(\pi)$ is simplified as f whenever there is no ambiguity. The following result provides the time complexity needed for solving problem $\boldsymbol{P}$.

Lemma 1. Problem $\boldsymbol{P}$ *can be solved in* $O(n^3)$ *time.*

Proof. Please refer to Appendix A.

During the processing process, however, machine breakdown may occur, which will disrupt the baseline schedule. Assume machine breakdown follows a given probability distribution that can be inferred from historical statistics, and let $\boldsymbol{B}$ be the stochastic variable to describe the breakdown time of machine. We also assume that once the machine fails, it will be immediately repaired. Considering the deterioration effect, the repair duration is modelled as a linear function of the breakdown time: $\gamma \boldsymbol{B} + M$ $(\gamma, M > 0)$, where M is the basic repair duration, and γ denotes the deteriorating rate. This reflects the practical situation where as the deterioration of the production condition accumulates, the time needed for repairing will usually become longer too [28]. The so-called *preempt-repeat* case is considered here, which means that if the machine breakdown occurs during the processing of a job, this job needs to be re-processed from the beginning. We also assume that the repairing can fully restore the initial state of the machine, meaning that the actual processing time of a job is reset to its normal processing time when it starts processing immediately after the completion of the repair.

Once repair is complete, it is desirable not to reschedule all remaining jobs to reduce computational burden and alleviate the negative impact on the entire production system resulting from the machine breakdown. In this work, we select a subset of subsequent unfinished jobs and fully compress them so that the repair duration can be compensated as soon as possible. As a result, the partially adjusted schedule will match up with the baseline schedule at a certain time point after the machine breakdown. In other words, from the match-up point on, the processing sequence and start-end time of each job will again become identical with the baseline schedule. The main criterion for the adjusted partial schedule, known as the rescheduling cost, is measured as the sum of the match-up time cost and the additional resource cost for the jobs to be further compressed. The rescheduling cost is formulated as follows:

$$g(\pi, \boldsymbol{B}) = mW_{min} + f_{W_{min}} \tag{3}$$

where W_{min} denotes the match-up time, m is the unit match-up time cost, and $f_{W_{min}}$ represents the additional resource cost for further compressing jobs' processing times. Since the unforeseen disruptions cause performance variability, the expectation of $g(\pi, \boldsymbol{B})$ denoted as $E(g(\pi, \boldsymbol{B}))$ is introduced as the robustness of the predictive schedule with stochastic machine breakdowns. Here, since no analytical function is available, the scenario-based evaluation approach is adopted to evaluate the solution robustness, which can be denoted as follows:

$$E(g(\pi, \boldsymbol{B})) = \frac{1}{s} \sum_{i=1}^{s} g(\pi, \boldsymbol{B_i}) \tag{4}$$

where s represents the size of the scenario set for scenario-based fitness evaluation of $E(g(\pi, \boldsymbol{B}))$, and $\boldsymbol{B_i}$ denotes the machine breakdown time of the ith scenario in the scenario set. $g(\pi, \boldsymbol{B})$ and $E(g(\pi, \boldsymbol{B}))$ are simplified as g and $E(g)$ or E separately when there is no ambiguity. Note that such simulation based evaluations of the solution robustness may become computationally very intensive and surrogate-assisted evolutionary techniques will be helpful.

A proactive baseline schedule is expected to be not only advantageous in operational cost, but also robust against stochastic machine breakdown. Hence, it would be ideal if we can find an optimal scheduling sequence and a resource allocation strategy that simultaneously minimizes the initial operational cost and robustness in response to machine breakdown. Unfortunately, such an ideal solution does not exist and typically, there is a tradeoff between minimal initial operational cost and solution robustness. That is, the baseline schedule having the minimal initial operational cost may not be optimal in terms of solution robustness, vice versa. As a result, we can only find a set of so-called Pareto optimal solutions, i.e., a set of schedules that present different tradeoffs between the two objectives. Again by using the representation scheme $\boldsymbol{\alpha}|\boldsymbol{\beta}|\boldsymbol{\gamma}$ [27], we formulate the above bi-objective scheduling problem denoted by $\boldsymbol{P1}$ as follows:

$$1 \left| \bar{p}_j + \alpha t_j - b_j u_j, brkdwn, rs \right| (E(g(\pi, \boldsymbol{B})),\ f(\pi)) \tag{5}$$

where '*brkdwn*' stands for the stochastic machine breakdown, and '*rs*' means robust scheduling.

4. A knowledge-based multi-objective evolutionary algorithm

As stochastic machine breakdown is considered in this work to solve the proactive scheduling problem, no traditional mathematical programming methods can be directly used due to the unavailability of explicit analytic formulations for its robust objective [29]. For this reason, we decide to design a multi-objective evolutionary algorithm (MOEA) for solving this problem, since MOEAs have been proved to be effective in solving many scheduling problems [49], including dynamic scheduling problems [30–32], steelmaking casting problems [48], knowledge-based reactive scheduling system development for emergency departments [50] and semiconductor final testing scheduling problems [51]. However, for evaluating the robustness of each solution, a large number of time-consuming Monte Carlo simulations have to be performed, since a set of stochastic machine breakdown scenarios need to be sampled. Therefore, we resort to surrogate-assisted evolutionary algorithms to reduce computation cost [17,33–35].

Surrogate-assisted evolutionary algorithms use surrogate models to replace computationally expensive real fitness evaluations, which have attracted increasing attention of researchers working on optimization of real-world applications involving expensive computation or physical simulation for evaluating the fitness of candidate solutions [38–41]. Many surrogate models have been employed to assist evolutionary algorithms, such as Gaussian process modeling, artificial neural networks, support vector regressions and radial basis functions, through which the computational cost can be reduced significantly due to the surrogate-assisted fitness evaluation instead of exact real fitness evaluation [34,36,37].

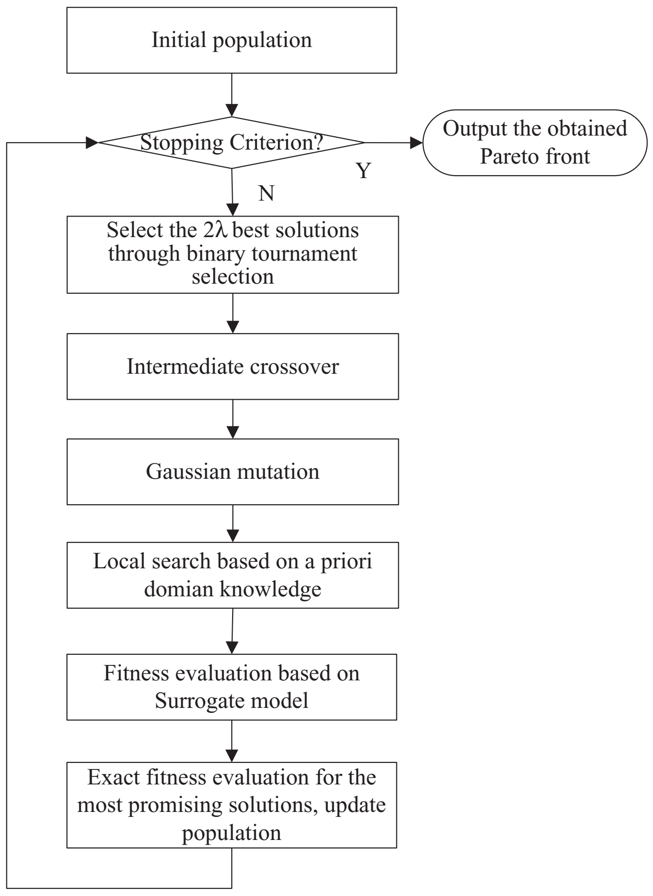
Initial population
Stopping Criterion?
Y
N
Output the obtained Pareto front
Select the 2λ best solutions through binary tournament selection
Intermediate crossover
Gaussian mutation
Local search based on a priori domian knowledge
Fitness evaluation based on Surrogate model
Exact fitness evaluation for the most promising solutions, update population

Fig. 1. Flow diagram of ADK/SA-NSGA-II.

In the following, we propose a knowledge-based, surrogate-assisted multi-objective evolutionary algorithm using elitist non-dominated sorting, termed ADK/SA-NSGA-II to solve the proactive scheduling problem. The original elitist non-dominated sorting genetic algorithm, known as NSGA-II [42], is a popular multi-objective evolutionary algorithm that has been successfully used for a variety of optimization problems. Within NSGA-II, we introduce a support vector regression mode as the surrogate to evaluate the solution robustness to replace the time-consuming simulations. To further enhance the search efficiency of the evolutionary search, the structural property based a priori domain knowledge is embedded in population initialization and offspring generation. The main steps of the proposed ADK/SA-NSGA-II algorithm are summarized in the following steps, also as illustrated in Fig. 1.

Step 1: Let $T = 1$, randomly sample *pop* solutions to form the initial population P_T, and evaluate each individual through simulation-based real fitness evaluation method.
Step 2: If the problem related stopping criterion is met, output the obtained Pareto front; otherwise go to Step 3.
Step 3: Select λ pairs of best solutions through tournament selection to form a parent population P_P, where λ is an integer with $\lambda \geq 1$ and is set as *pop*/2 by default.
Step 4: Apply intermediate crossover and Gaussian mutation on P_P to generate the offspring population P_O with 2λ individuals.
Step 5: Improve P_O through a priori domain knowledge based local search where the analytical structural property of problem ***P*1** is considered.
Step 6: Take the exact fitness values in the first *TrainScale* number of generations as the training data to prescreen P_O using support vector regression (SVR) model in the original space. The training data is updated in the following every *TrainingInternal* generations using the newly achieved 2λ exact fitness values, so is the SVR model.
Step 7: Evaluate the real fitness value of the estimated *SimulateRatio* percent best offspring from Step 6, and combine it with P_T to obtain P_{T+1}. Go back to Step 2.

The following remarks on the proposed ADK/SA-NSGA-II algorithm can be made.

(1) Simulation-based real fitness evaluations and fitness evaluations based on the surrogate are mainly for estimating the solution robustness $E(g)$ of a schedule for problem ***P*1** since no analytic formulations are available for $E(g)$. Efficient surrogate model is used to replace the computationally expensive real fitness evaluations, which can significantly reduce the computational cost. Here we assume that the computational cost for building the surrogates is negligible compared to the expensive simulation based fitness evaluations.
(2) We assume that the computational budget is limited and high quality solutions should be obtained given limited computational budget to meet the increasingly stringent industry requirements. Consequently, the stopping criterion is set as the number of simulation-based real fitness evaluations which is the product of maximum generation number *N*, exact evaluated offspring number in each generation *SimulateRatio* × *pop* and scenario set size *simulationTimes* for robustness evaluation.
(3) It is desirable to construct a good surrogate model for locating the most promising candidate solutions within a reasonable amount of computational effort. Although we assume that the computational cost of model construction is far less than that of the simulation-based real fitness evaluation, it is not realistic to use too many training data to train the surrogate model and reconstruct the model very often. On the other hand, using too few training data and insufficient reconstruction may deteriorate the reliability of the surrogate model and thus the solution quality.
(4) After the surrogate model is used to prescreen the offspring population, simulations will be used to re-evaluate part of population, which are typically the most promising solutions in the current population. The ratio of individuals to be re-evaluated is defined by a parameter *simulationRatio*. Re-evaluation of part of solutions can help prevent the search from being misled by the approximation errors of the surrogate models.
(5) All individuals in the first *TrainSetScale* generations of the evolution will be evaluated using the expensive simulations to collect data for training the surrogates. Once the surrogate is trained, it will replace the simulations to reduce computation time. In addition, all individuals in the last generation will also be evaluated using simulations to ensure that the fitness values of the Pareto optimal solutions obtained by the algorithm are correct.

4.1. Encoding scheme

The chromosome encoding the solutions to problem ***P*1** contains $2n$ elements, in which the first n elements form a permutation of n jobs and the following n elements correspond to a sequence of real numbers from interval [0, 1]. The encoding scheme is formulated as follows.

$$\vec{x}_{i,t} = \left(\overbrace{x_{i,1,t}, x_{i,2,t}, \ldots, x_{i,n,t}}^{\pi-part}, \underbrace{x_{i,n+1,t}, x_{i,n+2,t}, \ldots, x_{i,2n,t}}_{y-part} \right)$$
$$i = 1, 2, \ldots pop; t = 1, 2, \ldots, N \quad (6)$$

where *pop* stands for the population size, N stands for the maximum generation number, and $\vec{x}_{i,t}$ stands for the ith individual of the population in the tth generation. π-part of $\vec{x}_{i,t}$ directly encodes the sequence of all the jobs, and y-part presents the percentage of the pre-allocated resource to the corresponding job sequenced in π-part.

4.2. Main evolutionary operations

4.2.1. Intermediate crossover

With a probability of *CrossFraction*, the following intermediate crossover operation is applied on the π-parts and y-parts of two selected individuals indicated by subscripts $r1$ and $r2$, respectively.

$$o_{i,j,t} = p_{r1,j,t} + rand \cdot Ratio \cdot \left(p_{r2,j,t} - p_{r1,j,t} \right)$$
$$i = 1, 2, \cdots pop;\ j = 1, 2, \ldots, 2n;\ t = 1, 2, \ldots, N \tag{7}$$

where $o_{i,j,t}$ denotes the jth entry of the ith offspring individual in the tth generation, $p_{r1,j,t}$ and $p_{r2,j,t}$ represent the jth entries of two selected individuals in the tth generation. *rand* is a decimal randomly generated from [0,1], and *Ratio* is the parameter called 'ratio' to control the intermediate crossover process.

After adjusting each entries achieved through the above operation within its feasible interval, the π-part of each chromosome should be further repaired to fix infeasible solutions, for example those containing duplicated jobs in the sequence. The duplicated jobs should be replaced with missing ones.

4.2.2. Gaussian mutation

With a probability of *MutateFraction*, the following Gaussian mutation is applied on the π-part and y-part of the selected individual, respectively.

$$\tilde{o}_{i,j,t} = o_{i,j,t} + rand \cdot \left(Scale - Shrink \cdot Scale \cdot \frac{t}{N} \right) \cdot \left(u_{limit,j} - l_{limit,j} \right)$$
$$i = 1, 2, \ldots pop;\ j = 1, 2, \ldots, 2n;\ t = 1, 2, \ldots, N \tag{8}$$

where *rand* is a decimal, uniformly generated number within the interval [0, 1], *Scale* is used to control the mutation scale, *Shrink* represents the mutation shrink coefficient, $u_{limit,j}$ and $l_{limit,j}$ are the upper and lower bound of $o_{i,j,t}$. Repair should be applied on $\tilde{o}_{i,j,t}$ similar to the intermediate crossover operation to guarantee the feasibility of the solution.

4.2.3. Selection based on non-dominated sorting and crowing distance

After $E(g)$ and f are estimated for each individual, based on the dominance relationship between individuals, the current population is divided into K non-dominated fronts $\{PF_1, PF_2, \ldots, PF_K\}$. For any two fronts PF_i and PF_j, if $i < j$, then any individual in PF_j is dominated by some individual in PF_i. For individuals in the same front, a crowding distance is calculated to measure the degree of diversity of each individual. Fitness is assigned according to two criteria: individuals having a smaller front number are preferred in selection, and for individuals in the same front, those having a larger crowding distance will be prioritized. Refer to [42] for details about non-dominated sorting and crowding distance calculation.

4.3. Support vector regression model

4.3.1. Principles of support vector regression

It is assumed that a number of training data $\{(x_1, y_1), (x_2, y_2), \ldots, (x_l, y_l)\} \subset \mathbf{X} \times \mathbf{R}$ is available for constructing the support vector regression model, where l is the size of the training data, and $\mathbf{X}$ denotes the space of the input patterns $\mathbf{X} = \mathbf{R}^{\boldsymbol{d}}$. The main idea behind SVR is to map $x \in \mathbf{R}^d$ into some feature space by a non-linear mapping $\phi(\cdot)$, and to find a linear function $f(x)$ that has at most ε deviation from the actually obtained targets y_i for all the training data [43]. The linear function $f(x)$ is formulated as follows:

$$f(x) = \omega^T \phi(x) + b \tag{9}$$

where ω is the weight vector, and b is the threshold value.

In SVR method [43], the vector ω and the threshold b can be obtained via the following optimization:

$$\min \quad \frac{1}{2}\omega^2 + C\sum_{i=1}^{l}(\xi_i + \xi_i^*)$$
$$s.t. \quad \begin{cases} y_i - K(\omega, x_i) - b \le \varepsilon + \xi_i \\ K(\omega, x_i) + b - y_i \le \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \ \ge 0 \end{cases} \tag{10}$$

where C determines the tolerated deviation between the flatness of $f(x)$ and ε, $K(x_k, x) = \phi(x_k)^T \phi(x)$ is the kernel function. Here, the radial basis function is chosen as the kernel function $K(x_i, x_j)$, defined as $K(x_i, x_j) = e^{-\gamma x_i - x_j^2}$ [43].

4.3.2. Management of support vector regression model

By managing the SVR model, we mean the initial training of the surrogate in the beginning of the optimization and the control of the frequency of updating the SVR model during evolutionary optimization. In this work, we simply pre-specify that the surrogate is updated in every *TrainInternal* generations. If the surrogate is to be updated, the population in that generation is used to update the training data, and the newly generated data will be used to update the surrogate.

Step 1: Initialize the training data set with those individuals generated in the first *TrainSetScale* generations. All these individuals are exactly evaluated using the simulation-based real fitness evaluation method. Thus a training data set formulated is achieved as follows:

$$trainingSet = \{(x_{11}, x_{12}, \ldots, x_{1,2n}, E_1), \ldots, (x_{i1}, x_{i2}, \ldots, x_{i,2n}, E_i), \ldots, (x_{l1}, x_{l2}, \ldots, x_{l,2n}, E_l)\},\ i = 1, 2, \ldots, l$$

where $l = TrainSetScale \times pop$ denotes the row of the matrix, and $(x_{i1}, x_{i2}, \ldots, x_{i,2n}, E_i)$ represents the training data corresponding to the ith individual $(x_{i1}, x_{i2}, \ldots, x_{i,2n})$ with E_i as its robustness objective.

Step 2: Tune the two key parameters C and γ of the SVR model based on the training data using cross validation. The exponential grid interval for C is set as $[C_{min},\ C_{max}]$, and the moving step is set as C_{step}, meaning C takes a series of values $\{2^{C_{min}}, 2^{C_{min}+C_{step}}, \ldots, 2^{C_{max}}\}$. C_{min}, C_{max} and C_{step} are defaulted as -5, 5, 1, respectively. The alternative values for γ are similar with respect to $\gamma_{min}, \gamma_{max}$ and γ_{step} of the same default values. The training data is divided into three parts for cross validation, and the combination of C and γ with the most accurate estimation is selected.

Step 3: Construct the SVR model based on principles of support vector regression, where the training data obtained in Step 1 and the parameters tuned in Step 2 are adopted.

Step 4: After every *TrainInternal* number of generations, *trainingSet* should be updated by replacing the dominated individuals with those of higher qualities in the current generation. Consequently, the SVR model should be updated with the same method mentioned in its construction based on the updated training data and the tuned parameters.

4.3.3. SVR model based fitness evaluation

The constructed SVR model is embedded into the evolutionary algorithm to prescreen the obtained offspring population in each generation. Based on the evaluation results given by SVR model, *SimulateRatio* $\in$ [0, 1] percentage of the promising individuals are re-evaluated through simulation-based real fitness evaluation so as to prevent the search process from being misled to a false optimum [34,37].

4.4. Structural property based a priori domain knowledge

The following lemma defines the structure property that will be used as a priori domain knowledge of problem ***P*1** in ADK/SA-NSGA-II to enhance its search capability.

Lemma 2. *For a given resource allocation strategy, the objective of operational cost can be minimized by sequencing the jobs according to the non-decreasing order of* $\bar{p}_j - b_j u_j$.

Proof. Given the resource allocation amount for each job, the decision of sequencing jobs clearly does not affect the resource cost part of $f(\pi)$. For any two adjacent jobs violating the non-decreasing order of $\bar{p}_j - b_j u_j$, applying the pairwise exchange technique clearly reduce the total completion objective part. Therefore Lemma 2 holds.

According to Lemma 2, sequencing jobs in the non-decreasing order of $\bar{p}_j - b_j u_j$ at given resource allocation will improve the total completion time. Therefore, we apply this as a priori domain knowledge to the population initialization and part of the obtained offspring individuals by adjusting their π-parts to minimize their initial operational cost. Their y-parts are also adjusted accordingly to keep the sequence in consistency with the π-parts.

5. Comparative studies

5.1. Experimental design

In order to examine the efficiency of the proposed knowledge-based surrogate-assisted multi-objective evolutionary algorithm (ADK/SA-NSGA-II), ten problem cases of randomly generated numerical instances are considered. The problem cases are categorized based on the number of jobs, and labeled as Cases 1 to 10 for 30, 50, 70, 90, 110, 130, 150, 200, 300 and 500 jobs, respectively. For each case, 100 numerical test instances are randomly generated, resulting in a total of 1000 test instances. All the algorithms are implemented in MATLAB and run on a PC with 4 G RAM, Intel Core i5 CPU 2.5 GHz. In each problem case, the parameter settings are as follows:

(1) The normal processing time $\bar{p}_j$, positive compression rate b_j, unit resource cost c_j of each job are all decimals randomly generated from different uniform distributions $U[1, 100]$, $U[0, 1]$ and $U[2, 9]$, respectively, where $j = 1, 2, \ldots, n$. The maximum available resource amount $\bar{u}_j$ of each job is a decimal randomly generated from the corresponding uniform distribution $U[0.6\bar{p}_j/b_j, 0.9\bar{p}_j/b_j]$.
(2) The deterioration rate is set as $\alpha = 0.01$, the parameter of the exponential distribution of machine breakdown is set as $E(\boldsymbol{B}) = 100$, the basic maintenance duration is set as $M = 30$, the deterioration rate of repairing is set as $\gamma = 0.01$, the unit completion time cost is set as $q = 1$ and the unit match-up time cost is set as $m = 100$ according to the empirical data in practice.
(3) The size of scenario set is set as $SimulationTimes = 30$ to evaluate the solution robustness of the candidate schedule.

In order to examine the performance of the proposed ADK/SA-NSGA-II, two other NSGA-II based algorithms are compared on the above test instances. One is NSGA-II using simulations only for fitness evaluations (denoted as NSGA-II), the other is NSGA-II assisted by a SVR model for fitness evaluations, which is termed as SA-NSGA-II. The encoding schemes and evolutionary operators of NSGA-II and SA-NSGA-II are same as those of ADK/SA-NSGA-II for the fairness of the comparison. We first compare SA-NSGA-II with NSGA-II to examine the efficiency of the proposed SVR model, and compare ADK/SA-NSGA-II with SA-NSGA-II to understand the improvement offered by a priori domain knowledge. In addition, we compare our algorithms with MOEA/D [46,47], another popular multi-objective evolutionary algorithm, which has been successfully applied to solve various discrete and continuous multi-objective problems [46,47]. The encoding schemes and evolutionary operators of MOEA/D are also the same as those of ADK/SA-NSGA-II for the fairness of the comparison.

To understand the search capabilities of the four algorithms under comparison, we will present the initial population, convergence curves and the obtained Pareto front of each algorithm from one typical run, respectively. For benchmarking, we also apply the assignment model proposed in Lemma 1 to solve the test instance. This solution will serve as a reference, which only minimizes the operational cost of the problem.

Table 1
The results of parameters tuning.

Parameters	Options	Distances	Parameters	Options	Distances
pop	50	3154	*N*	150	2069
	100	1862		**200**	1853
	150	2037		250	1853
TrainSetScale	1	4876	*MutateFraction*	$1/n$	2760
	3	4429		$\boldsymbol{2/n}$	2163
	5	2123		$3/n$	3053
	7	3101		$4/n$	4181
TrainInternal	3	4622	*Ratio*	0.8	3368
	5	5712		**1.2**	2529
	7	4475		1.6	3351
	9	5430	*(Scale,Shrink)*	(0.1,0.1)	5065
SimulateRatio	0.3	2014		(0.1,0.5)	5885
	0.5	1981		(0.1,0.9)	8128
	0.7	2793		(0.5,0.1)	2423
	0.9	3786		**(0.5,0.5)**	1732
CrossFraction	$\boldsymbol{1/n}$	2628		(0.5,0.9)	2787
	$2/n$	2739		(0.9,0.1)	2089
	$3/n$	3790		(0.9,0.5)	2171
	$4/n$	3326		(0.9,0.9)	2217

5.2. Parameters tuning

The parameters of ADK/SA-NSGA-II algorithm, such as population size *pop*, the maximum number of generations *N*, SVR model related parameters (*TrainSetScale, TrainInternal*, and *SimulateRatio*) and evolutionary operation related parameters (*CrossFraction, Ratio, MutateFraction, Scale* and *Shrink*) will certainly influence its performance. Therefore, parameters are first tuned based on the performance of the results from 50 runs on Case 2. The average minimum distance of the obtained Pareto front to the ideal point is used as an indicator for the performance of each tuned value of the parameter, where the ideal point is constructed by the minimal operational cost of the predictive schedule and the lower bound of the robustness objective. The tuned results are shown in Table 1, and the highlighted values are chosen for the subsequent comparisons. NSGA-II, MOEA/D and SA-NSGA-II share the same parameter values with ADK/SA-NSGA-II if needed to ensure fair comparison. In MOEA/D, the number of the weight vectors in the neighborhood of each weight vector *T* is set to be 30 based on the results of our pilot studies.

Note that $N = 200$ is the tuned value of the maximum number of generations. The corresponding computational cost of ADK/SA-NSGA-II measured by the number of fitness evaluations using simulations can be calculated as $N \times pop \times SimulateRatio \times SimulationTimes$, where $pop \times SimulateRatio \times SimulationTimes$ is the number of simulations employed in each generation. For fair comparison, the maximum number of generations of NSGA-II and MOEA/D is set as $N \times SimulateRatio$, so that the number of expensive

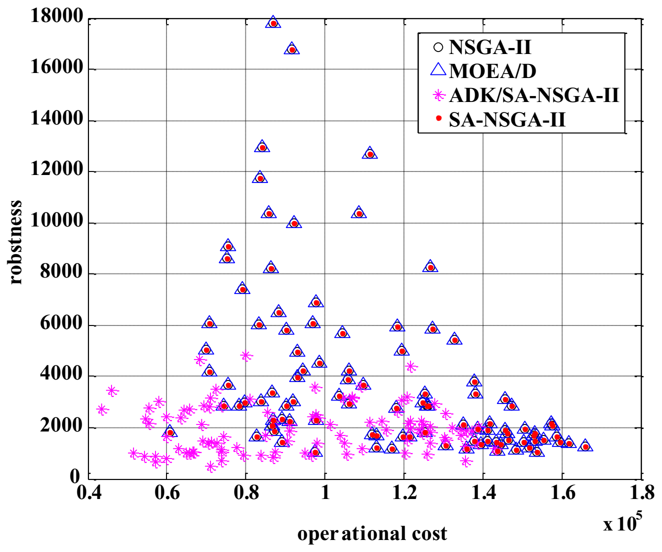

Fig. 2. The initial populations of NSGA-II, MOEA/D, SA-NSGA-II and ADK/SA-NSGA-II.

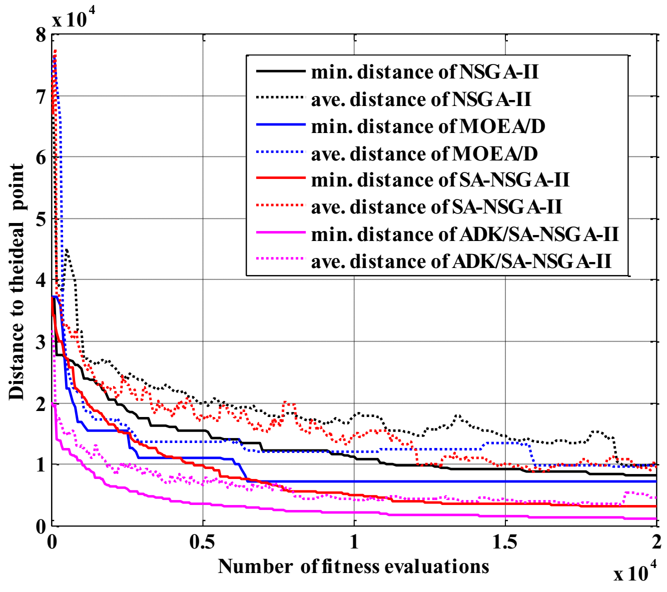

Fig. 3. The convergence processes of NSGA-II, MOEA/D, SA-NSGA-II and ADK/SA-NSGA-II.

simulations it uses is the same as ADK/SA-NSGA-II. And SA-NSGA-II takes the same maximum number of generations as ADK/SA-NSGA-II for its surrogate-assisted fitness evaluations.

5.3. Results

5.3.1. Optimization process of one typical test instance

The initial populations, convergence processes and the obtained Pareto fronts of the four compared algorithms in solving one typical test instance of Case 2 are shown in Figs. 2, 3 and 4, respectively. We use these as examples of illustrating the contribution of SVR model and the improvement gained by incorporating domain knowledge in evolution. The results of Cases 1, 3–10 are similar to Case 2, and therefore are not shown here. From Fig. 2, we can observe that SA-NSGA-II, NSGA-II and MOEA/D have the same initial population, and the initial population of ADK/SA-NSGA-II is improved owing to the adjustment done by incorporating a priori domain knowledge.

The convergence processes of NSGA-II, MOEA/D, SA-NSGA-II and ADK/SA-NSGA-II under the same computational cost are shown in

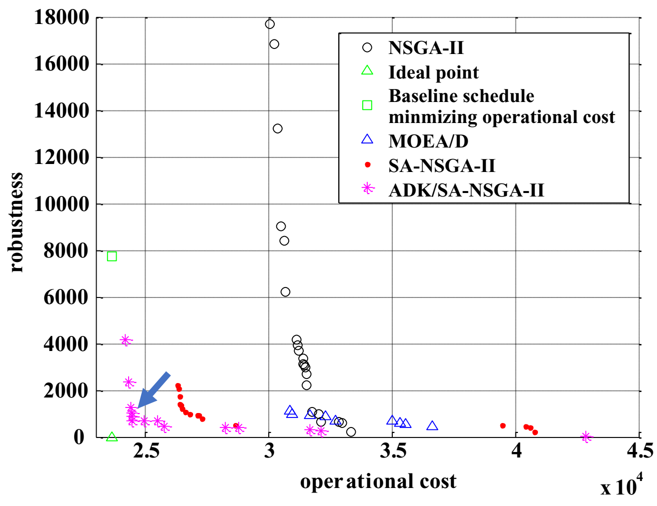

Fig. 4. The obtained Pareto fronts of the four algorithms.

Fig. 3. The convergence curves show that fitness values converge rapidly in the four algorithms from the beginning to 5000 simulation-based fitness evaluations. SA-NSGA-II outperforms NSGA-II with any number of exact fitness evaluations. MOEA/D is competitive with respect to SA-NSGA-II in the early stage of the search. However, as the evolution continues, the advantage of SA-NSGA-II becomes more obvious. These indicate the efficiency of the constructed SVR model in speeding up the convergence. The convergence curves of ADK/SA-NSGA-II shows clear advantages over NSGA-II, MOEA/D and SA-NSGA-II during the entire optimization process, which shows that a priori domain knowledge leads to great improvement in the convergence speed of the proposed algorithm.

Fig. 4 shows the obtained Pareto fronts from the four compared algorithms. It is observed that SA-NSGA-II achieves a better Pareto front in comparison with NSGA-II. MOEA/D achieves similar robustness objectives to that of SA-NSGA-II, but has a much larger operational cost on average. These indicate that the SVR model can efficiently improve the operational cost of the schedules as well as their robustness in response to machine breakdown. By comparing the Pareto fronts approximated by ADK/SA-NSGA-II and SA-NSGA-II, we can see the Pareto optimal solutions obtained by ADK/SA-NSGA-II are further improved. Based on these observations, we conclude that a priori domain knowledge has significantly enhanced the searching capability of the evolutionary algorithm.

By examining the optimal solution obtained by solving problem ***P*** through the assignment model proposed in Lemma 1 and those Pareto optimal solutions approximated by ADK/SA-NSGA-II, we can see that the robustness of the solution will be very poor if only the operational cost is minimized in response to machine breakdown. It is indeed very helpful to make use of information about machine breakdown to find non-dominated solutions that also take schedule robustness into account. From the Pareto optimal solutions obtained by the proposed ADK/SA-NSGA-II, it is very straightforward for the decision-maker to pick out a solution that has an acceptable operational cost but considerably reduced rescheduling cost, for example the one indicated by the arrow in Fig. 4.

5.3.2. Quantitative comparisons

Convergence and diversity of the obtained Pareto optimal solutions are two most important aspects for measuring the quality of the solutions. Among many others, inverted generational distance (*IGD*) and hyper-volume (*HV*) are two popular performance indicators. *IGD* measures the average distance between the obtained Pareto front denoted by *PF* to the optimal Pareto front denoted by *PF*∗ [44]:

$$IGD(PF^*, PF) = \frac{\sum_{v \in PF^*} d(v, PF)}{|PF^*|} \quad (11)$$

Table 2
Comparative results between SA-NSGA-II and NSGA-II.

		IGD		HV	
		Ave	Std	Ave	Std
Case 1	NSGA-II	0.04	0.03	2.19×10^7	3.14×10^7
	SA-NSGA-II	0.01	0.01	2.22×10^7	3.14×10^7
Case 2	NSGA-II	0.08	0.06	3.77×10^9	3.04×10^{10}
	SA-NSGA-II	0.02	0.03	3.78×10^9	3.04×10^{10}
Case 3	NSGA-II	0.15	0.09	9.19×10^7	2.29×10^8
	SA-NSGA-II	0.01	0.01	1.11×10^8	2.79×10^8
Case 4	NSGA-II	0.26	0.13	1.92×10^8	1.18×10^9
	SA-NSGA-II	0.01	0.02	2.33×10^8	1.35×10^9
Case 5	NSGA-II	0.34	0.15	1.40×10^8	6.30×10^8
	SA-NSGA-II	0.00	0.01	2.29×10^8	1.00×10^9
Case 6	NSGA-II	0.46	0.20	6.93×10^7	1.77×10^8
	SA-NSGA-II	0.01	0.03	1.39×10^8	2.25×10^8
Case 7	NSGA-II	0.44	0.18	1.43×10^8	3.08×10^8
	SA-NSGA-II	0.01	0.03	2.77×10^8	4.33×10^8
Case 8	NSGA-II	0.60	0.16	1.14×10^9	7.49×10^9
	SA-NSGA-II	0.00	0.01	2.52×10^9	1.04×10^{10}
Case 9	NSGA-II	0.69	0.14	7.84×10^8	8.32×10^8
	SA-NSGA-II	0.00	0.00	4.35×10^9	2.20×10^9
Case 10	NSGA-II	0.74	0.10	2.83×10^{10}	2.51×10^{10}
	SA-NSGA-II	0.00	0.00	2.37×10^{11}	9.74×10^{10}

Table 3
Comparative results between MOEA/D and NSGA-II.

		IGD		HV	
		Ave	Std	Ave	Std
Case 1	MOEA/D	0.13	0.06	1.20×10^7	1.85×10^7
	NSGA-II	0.00	0.00	1.49×10^7	2.12×10^7
Case 2	MOEA/D	0.08	0.06	3.76×10^9	3.03×10^{10}
	NSGA-II	0.01	0.02	3.76×10^9	3.03×10^{10}
Case 3	MOEA/D	0.03	0.05	9.03×10^7	1.99×10^8
	NSGA-II	0.13	0.12	9.25×10^7	2.21×10^8
Case 4	MOEA/D	0.01	0.01	4.76×10^8	3.26×10^9
	NSGA-II	0.29	0.17	4.58×10^8	3.27×10^9
Case 5	MOEA/D	0.00	0.00	2.80×10^8	1.06×10^9
	NSGA-II	0.42	0.16	1.79×10^8	6.76×10^8
Case 6	MOEA/D	0.00	0.01	1.45×10^8	1.24×10^8
	NSGA-II	0.60	0.16	4.07×10^7	6.65×10^7
Case 7	MOEA/D	0.00	0.00	3.66×10^8	4.80×10^8
	NSGA-II	0.61	0.15	1.26×10^8	3.03×10^8
Case 8	MOEA/D	0.00	0.00	3.92×10^9	1.30×10^{10}
	NSGA-II	0.75	0.13	1.21×10^9	7.51×10^9
Case 9	MOEA/D	0.00	0.00	1.21×10^{10}	8.66×10^9
	NSGA-II	0.85	0.07	9.00×10^8	2.05×10^9
Case 10	MOEA/D	0.00	0.00	7.54×10^{11}	2.10×10^{11}
	NSGA-II	0.89	0.05	2.77×10^{10}	2.53×10^{10}

Table 4
Comparative results between ADK/SA-NSGA-II and MOEA/D.

		IGD		HV	
		Ave	Std	Ave	Std
Case 1	ADK/SA-NSGA-II	0.00	0.00	1.23×10^7	2.37×10^7
	MOEA/D	0.25	0.09	9.00×10^6	2.07×10^7
Case 2	ADK/SA-NSGA-II	0.00	0.01	4.46×10^8	3.33×10^9
	MOEA/D	0.36	0.14	3.91×10^8	2.98×10^9
Case 3	ADK/SA-NSGA-II	0.01	0.02	6.40×10^7	1.96×10^8
	MOEA/D	0.40	0.16	3.11×10^7	1.19×10^8
Case 4	ADK/SA-NSGA-II	0.02	0.04	1.02×10^8	4.19×10^8
	MOEA/D	0.39	0.21	7.25×10^7	3.67×10^8
Case 5	ADK/SA-NSGA-II	0.04	0.10	1.37×10^8	5.38×10^8
	MOEA/D	0.28	0.22	7.79×10^7	2.65×10^8
Case 6	ADK/SA-NSGA-II	0.15	0.18	4.64×10^7	9.48×10^7
	MOEA/D	0.14	0.19	5.60×10^7	1.38×10^8
Case 7	ADK/SA-NSGA-II	0.29	0.20	4.46×10^7	7.16×10^7
	MOEA/D	0.04	0.11	6.69×10^7	6.63×10^7
Case 8	ADK/SA-NSGA-II	0.50	0.24	4.36×10^8	2.05×10^9
	MOEA/D	0.01	0.05	6.19×10^8	2.31×10^9
Case 9	ADK/SA-NSGA-II	0.50	0.22	9.22×10^8	3.08×10^9
	MOEA/D	0.01	0.06	1.95×10^9	3.00×10^9
Case 10	ADK/SA-NSGA-II	0.02	0.07	4.68×10^{10}	3.89×10^{10}
	MOEA/D	0.50	0.29	1.19×10^{10}	1.08×10^{10}

Table 5
Comparative results between ADK/SA-NSGA-II and SA-NSGA-II.

		IGD		HV	
		Ave	Std	Ave	Std
Case 1	ADK/SA-NSGA-II	0.02	0.02	1.85×10^7	3.00×10^7
	SA-NSGA-II	0.07	0.05	1.81×10^7	3.14×10^7
Case 2	ADK/SA-NSGA-II	0.02	0.02	4.86×10^8	3.34×10^9
	SA-NSGA-II	0.17	0.09	4.65×10^8	3.21×10^9
Case 3	ADK/SA-NSGA-II	0.02	0.04	6.55×10^7	1.66×10^8
	SA-NSGA-II	0.23	0.14	4.64×10^7	1.26×10^8
Case 4	ADK/SA-NSGA-II	0.01	0.02	1.20×10^8	5.56×10^8
	SA-NSGA-II	0.32	0.19	7.93×10^7	3.67×10^8
Case 5	ADK/SA-NSGA-II	0.02	0.04	7.44×10^7	8.60×10^7
	SA-NSGA-II	0.32	0.19	4.49×10^7	6.32×10^7
Case 6	ADK/SA-NSGA-II	0.02	0.05	1.01×10^8	1.69×10^8
	SA-NSGA-II	0.32	0.22	8.00×10^7	1.77×10^8
Case 7	ADK/SA-NSGA-II	0.02	0.05	1.45×10^8	2.53×10^8
	SA-NSGA-II	0.35	0.22	7.37×10^7	1.13×10^8
Case 8	ADK/SA-NSGA-II	0.02	0.06	3.13×10^8	3.76×10^8
	SA-NSGA-II	0.41	0.21	1.59×10^8	2.34×10^8
Case 9	ADK/SA-NSGA-II	0.00	0.00	3.57×10^9	2.20×10^9
	SA-NSGA-II	0.70	0.13	5.52×10^8	5.64×10^8
Case 10	ADK/SA-NSGA-II	0.00	0.00	4.18×10^{11}	1.27×10^{11}
	SA-NSGA-II	0.88	0.07	1.68×10^{10}	1.58×10^{10}

where v represents a Pareto solution in the obtained Pareto front PF, $d(v, PF)$ represents the distance between v and PF, and $|PF*|$ denotes the number of different Pareto solutions in $PF*$, which is either a representative set chosen from a known theoretical Pareto, or a set of non-dominated solutions selected from the combination of all Pareto fronts to be compared, if the theoretical Pareto front is unknown. The *IGD* metric mainly reflects the convergence of a Pareto front, and a smaller value indicates better convergence. The *HV* metric calculates the area dominated by the obtained Pareto front [45], which can account for both convergence and diversity of the obtained solution set, referring to [45] for details. The larger the *HV* value, the better the performance.

The results of the four algorithms in terms of *IGD* and *HV* are presented in Tables 2, 3, 4, 5, respectively. The results are averaged over 100 independently generated test instances for each case. From Table 2, we can see that the mean *IGD* values of SA-NSGA-II are lower than those of NSGA-II on all ten problem cases with a smaller or similar standard deviation, and the mean *HV* values of SA-NSGA-II are also better than those of NSGA-II. The improvement in *IGD* metric becomes more obvious as the problem size increases. Thus, it is evident that SA-NSGA-II outperforms NSGA-II, demonstrating that the use of surrogates can efficiently improve the performance. This improvement becomes more significant as the problem size increases.

Table 3 indicates that NSGA-II has a better performance in terms of convergence and solution diversity when the problem size is relatively small (30 and 50 jobs). As the problem size increases to 500 jobs, MOEA/D becomes more competitive with better mean *IGD* and *HV* values. However, ADK/SA-NSGA-II can still outperform MOEA/D, as shown in Table 4.

From Table 5, we can see that the *IGD* values of ADK/SA-NSGA-II are much better than SA-NSGA-II. So are the *HV* values. The improvement of the *IGD* and the *HV* values also becomes more obvious with the increase of problem size. Consequently, we can conclude that ADK/SA-NSGA-II outperforms SA-NSGA-II thanks to the incorporation of a priori domain knowledge for guiding the search process. The improvement resulting from the embedding of a priori domain knowledge becomes more significant as the problem size increases, which implies that such domain knowledge will be increasingly helpful in solving large scheduling problems.

Table 6
Comparison of relative runtime of the four algorithms.

Cases	Compared algorithms	RTI
Case 1	NSGA-II & MOEA/D	−0.33
	ADK/SA-NSGA-II & SA-NSGA-II	−0.05
Case 2	NSGA-II & MOEA/D	−0.34
	ADK/SA-NSGA-II & SA-NSGA-II	−0.02
Case 3	NSGA-II & MOEA/D	−0.33
	ADK/SA-NSGA-II & SA-NSGA-II	−0.03
Case 4	NSGA-II & MOEA/D	−0.33
	ADK/SA-NSGA-II & SA-NSGA-II	−0.03
Case 5	NSGA-II & MOEA/D	−0.34
	ADK/SA-NSGA-II & SA-NSGA-II	−0.06
Case 6	NSGA-II & MOEA/D	−0.32
	ADK/SA-NSGA-II & SA-NSGA-II	−0.08
Case 7	NSGA-II & MOEA/D	−0.33
	ADK/SA-NSGA-II & SA-NSGA-II	−0.04
Case 8	NSGA-II & MOEA/D	−0.34
	ADK/SA-NSGA-II & SA-NSGA-II	−0.10
Case 9	NSGA-II & MOEA/D	−0.36
	ADK/SA-NSGA-II & SA-NSGA-II	−0.12
Case 10	NSGA-II & MOEA/D	−0.39
	ADK/SA-NSGA-II & SA-NSGA-II	−0.17

The runtime comparisons between NSGA-II, MOEA/D, SA-NSGA-II and ADK/SA-NSGA-II are presented in Table 6. Each reported value denotes a relative time indicator (RTI) calculated by $(RT_1 - RT_2)/RT_2$, where RT_1 and RT_2 represent the runtime of the two compared algorithms, respectively. So a negative value indicates the first algorithm is faster. From these results, we can see that NSGA-II takes less time than MOEA/D, and ADK/SA-NSGA-II takes less time than SA-NSGA-II. These results demonstrate that the proposed algorithm ADK/SA-NSGA-II is effective in reducing computational time.

5.3.3. The ANOVA results

The analysis of variance (ANOVA) is implemented using the commercial software SPSS (Version 17.0) to analyze the *IGD* and *HV* values obtained by the compared algorithms. The resulting *F*-ratios and *p*-values of ANOVA are shown in Tables 7 and 8. The difference is considered to be statistically significant if the *p*-value is less than 0.05. From Tables 7 and 8, we can draw the following conclusions. First, for the *IGD* measure, the algorithm factor has a statistically significant impact for nearly all cases. ADK/SA-NSGA-II performs significantly better than MOEA/D and SA-NSGA-II in terms of convergence. This is also true for MOEA/D against NSGA-II, and SA-NSGA-II against NSGA-II. Second, for the *HV* measure, the impact of the algorithm factor is not significant for small cases with 30 to 200 jobs (Cases 1–8). But the impact becomes significant for 300 and 500 jobs (Cases 9 and 10), indicating that improvement in computational efficiency achieved by the proposed algorithm is more significant for solving larger problems.

6. Conclusions

In this paper, we address the proactive scheduling problem in the presence of stochastic machine breakdown in the deteriorating production environments. A knowledge-based multi-objective evolutionary algorithm is proposed to solve the problem, where support vector regression based surrogate models are employed to reduce the computation cost resulting from the extra time-consuming simulations for evaluating the solution robustness, and analytical a priori domain knowledge is introduced to guide the searching process. Comparative results clearly show that both surrogate-assisted techniques and analytical a priori domain knowledge are effective in enhancing the time efficiency and search capability of the evolutionary algorithm. Our algorithm is very generic and applicable to other computationally expensive black-box optimization problems, such as aerodynamic design optimization problems and energy performance optimization of building, just to name a few.

The proposed algorithm has been shown to be efficient and promising in handling stochastic machine breakdown, where the SVR-based surrogate model plays an important role in reducing the computational cost resulting from the simulation-based fitness evaluations. On the other hand, we note that NSGA-II is outperformed by

Table 7
The ANOVA result for the *IGD* performance measure.

Cases	SA-NSGA-II & NSGA-II		MOEA/D & NSGA-II		ADK/SA-NSGA-II & MOEA/D		ADK/SA-NSGA-II & SA-NSGA-II	
	F-ratio	*p*-value	*F*-ratio	*p*-value	*F*-ratio	*p*-value	*F*-ratio	*p*-value
Case 1	23.37	0.000	371.19	0.000	510.69	0.000	61.10	0.000
Case 2	55.74	0.000	64.37	0.000	420.05	0.000	184.40	0.000
Case 3	145.33	0.000	39.67	0.000	394.89	0.000	146.88	0.000
Case 4	238.95	0.000	186.22	0.000	195.52	0.000	176.30	0.000
Case 5	333.95	0.000	426.76	0.000	62.25	0.000	163.62	0.000
Case 6	332.81	0.000	931.21	0.000	0.14	0.705	115.47	0.000
Case 7	361.71	0.000	1142.45	0.000	84.34	0.000	141.98	0.000
Case 8	902.40	0.000	2252.67	0.000	265.66	0.000	211.50	0.000
Case 9	1579.44	0.000	9822.67	0.000	297.20	0.000	1796.25	0.000
Case 10	3389.11	0.000	18097.48	0.000	166.47	0.000	9401.54	0.000

Table 8
The ANOVA result for the *HV* performance measure.

Cases	SA-NSGA-II & NSGA-II		MOEA/D & NSGA-II		ADK/SA-NSGA-II & MOEA/D		ADK/SA-NSGA-II & SA-NSGA-II	
	F-ratio	*p*-value	*F*-ratio	*p*-value	*F*-ratio	*p*-value	*F*-ratio	*p*-value
Case 1	0.00	0.96	0.70	0.403	0.73	0.395	0.00	0.947
Case 2	0.00	1.00	0.00	0.999	0.01	0.921	0.00	0.970
Case 3	0.19	0.67	0.00	0.950	1.35	0.247	0.56	0.457
Case 4	0.03	0.85	0.00	0.974	0.19	0.667	0.25	0.619
Case 5	0.38	0.54	0.43	0.512	0.63	0.427	5.07	0.026
Case 6	3.95	0.05	36.63	0.000	0.21	0.644	0.50	0.480
Case 7	4.21	0.04	11.84	0.001	3.44	0.066	4.40	0.038
Case 8	0.76	0.38	2.13	0.146	0.23	0.631	7.99	0.005
Case 9	151.12	0.00	105.35	0.000	3.78	0.004	116.99	0.000
Case 10	284.18	0.00	778.33	0.000	49.04	0.000	650.05	0.000

MOEA/D, indicating that NSGA-II may be replaced by other state-of-the-art multi-objective evolutionary algorithms. Our future research directions will focus on the following aspects. First, other robustness measures should be considered in the proactive scheduling to find a more efficient and effective way to deal with uncertainties. Second, it is worthy of investigating the effectiveness of surrogates other than the SVR model used in this work. More sophisticated surrogate management techniques should also be developed to further improve the efficiency of the surrogate-assisted evolutionary algorithms. Third, mathematical analysis using dynamic system theory, such as Markov chains will be used to prove and explain the convergence of the proposed method. Last but not the least, the proposed surrogate method will be embedded into other state-of-the-art multi-objective evolutionary algorithms, such as MOEA/D and the inverse model based multi-objective evolutionary algorithm (IM-MOEA) [52]. The use of surrogates in other metaheuristics, such as firefly algorithms, cuckoo search, bat algorithms, Krill Herd and monarch butterfly optimization (MBO), will also be explored.

Acknowledgments

This research was supported in part by the National Natural Science Foundation of China (grant nos. 71420107028, 71501024, 71502026, 71533001) and by Fundamental Research Funds for the Central Universities under grant DUT15QY32.

Appendix A. Proof of Lemma 1

Proof. An optimal solution of problem $\boldsymbol{P}$ can be determined by the following steps:

(1) For job sequence $S = (J_{[1]}, J_{[2]}, \ldots, J_{[n]})$, the total operational cost can be formulated as follows:

$$\begin{aligned} q\sum_{j=1}^{n} C_j + \sum_{j=1}^{n} c_j u_j &= q\sum_{j=1}^{n} C_{[j]} + \sum_{j=1}^{n} c_{[j]} u_{[j]} \\ &= q(C_{[1]} + C_{[2]} + \cdots + C_{[n]}) + \sum_{j=1}^{n} c_{[j]} u_{[j]} \\ &= q\sum_{j=1}^{n} (n+1-j) p_{[j]} + \sum_{j=1}^{n} c_{[j]} u_{[j]} \\ &= q\sum_{j=1}^{n} \tilde{w}_j p_{[j]} + \sum_{j=1}^{n} c_{[j]} u_{[j]} \end{aligned}$$

where $\tilde{w}_j = n + 1 - j$ for $j = 1, 2, \ldots, n$ denotes the position weight, $[j]$ represents the index of the job arranged in the jth position of the schedule.

Now substituting $p_j = \bar{p}_j + \alpha t_j - b_j u_j$ into the above equation, we have

$$\begin{aligned} q\sum_{j=1}^{n} \tilde{w}_j p_{[j]} + \sum_{j=1}^{n} c_{[j]} u_{[j]} &= q\sum_{j=1}^{n} W_j \left(\bar{p}_{[j]} - b_{[j]} u_{[j]}\right) + \sum_{j=1}^{n} c_{[j]} u_{[j]} \\ &= q\sum_{j=1}^{n} W_j \bar{p}_{[j]} + \sum_{j=1}^{n} \left(c_{[j]} - qW_j b_{[j]}\right) u_{[j]} \end{aligned}$$

where $W_j = \sum_{i=j}^{n} \sum_{k=0}^{i-j} \mathbf{M}_{i-j+1,k+1} \alpha^k \tilde{w}_i$, and $\mathbf{M}_{i,j}$ can be iteratively calculated as $\mathbf{M}_{i,j} = \mathbf{M}_{i-1,j-1} + \mathbf{M}_{i-1,j}$ with $\mathbf{M}_{1,1} = 1$ $\mathbf{M}_{1,j} = 0$, $\mathbf{M}_{i,1} = 0$ for $i, j = 2, 3, \ldots, n$.

Therefore for the jth job in given schedule, when $c_{[j]} \geq qW_j b_{[j]}$ its optimal resource allocation amount should be $u_{[j]} = 0$, otherwise, its optimal resource allocation amount should be $u_{[j]} = \bar{u}_{[j]}$.

(2) For $1 \leq j, r \leq n$, let us define

$$C_{jr} = \begin{cases} W_r \bar{p}_j, & r = 1, 2, \ldots, n;\ c_j \geq qW_r b_j \\ W_r \bar{p}_j + \left(c_j - qW_r b_j\right)\bar{u}_j, & r = 1, 2, \ldots, n;\ c_j < qW_r b_j \end{cases}$$

where C_{jr} denotes the minimum possible cost resulting from assigning job J_j to position r in the sequence.

Here we introduce binary variable z_{jr} to denote whether job J_j is arranged at the rth position in the schedule: $z_{jr} = 1$ means 'yes' and $z_{jr} = 0$ means 'no'. Then the problem $\boldsymbol{P}$ can be transferred into the following assignment problem:

$$\begin{aligned} &\min \sum_{r=1}^{n} \sum_{j=1}^{n} C_{jr} z_{jr} \\ &\text{s.t.} \sum_{r=1}^{n} z_{jr} = 1,\ j = 1, 2, \ldots, n \\ &\sum_{j=1}^{n} z_{jr} = 1,\ r = 1, 2, \ldots, n \\ &z_{jr} = 1 \text{ or } z_{jr} = 0,\ r, j = 1, 2, \ldots, n \end{aligned}$$

References

[1] M.L. Pinedo, Scheduling: Theory Algorithms and Systems, Springer Science+Business Media, New York, 2008.
[2] S. Goren, I. Sabuncuoglu, Robustness and stability measures for scheduling: single-machine environment, IIE Trans 40 (1) (2008) 66–83.
[3] M. Sevaux, K.S. Rensen, A genetic algorithm for robust schedules in a one-machine environment with ready times and due dates, 4OR 2 (1) (2004) 129–147.
[4] S. Goren, I. Sabuncuoglu, Optimization of schedule robustness and stability under random machine breakdowns and processing time variability, IIE Trans. 42 (3) (2010) 203–220.
[5] W. Herroelen, R. Leus, Robust and reactive project scheduling: a review and classification of procedures, Int. J. Prod. Res. 42 (8) (2004) 1599–1620.
[6] H. Aytug, M.A. Lawley, K. McKay, S. Mohan, R. Uzsoy, Executing production schedules in the face of uncertainties: a review and some future directions, Eur. J. Oper. Res. 161 (1) (2005) 86–110.
[7] I. Sabuncuoglu, S. Goren, Hedging production schedules against uncertainty in manufacturing environment with a review of robustness and stability research, Int. J. Comput. Integr. Manuf. 22 (2) (2009) 138–157.
[8] S.V. Mehta, R.M. Uzsoy, Predictable scheduling of a job shop subject to breakdowns, IEEE Rob. Autom. Mag 14 (3) (1998) 365–378.
[9] R. Leus, W. Herroelen, Scheduling for stability in single-machine production systems, J. Sched. 10 (3) (2007) 223–235.
[10] B. Yang, J. Geunes, Predictive-reactive scheduling on a single resource with uncertain future jobs, Eur. J. Oper. Res. 189 (3) (2008) 1267–1283.
[11] R. O'Donovan, R. Uzsoy, K.N. McKay, Predictable scheduling of a single machine with breakdowns and sensitive jobs, Int. J. Prod. Res. 37 (18) (1999) 4217–4233.
[12] L. Liu, H.Y. Gu, Y.G. Xi, Robust and stable scheduling of a single machine with random machine breakdowns, Int. J. Adv. Manuf. Technol. 31 (7–8) (2007) 645–654.
[13] D. Briskorn, J. Leung, M.L. Pinedo, Robust scheduling on a single machine using time buffers, IIE Trans 43 (6) (2011) 383–398.
[14] D. Shabtay, G. Steiner, A survey of scheduling with controllable processing times, Discret, Appl. Math 155 (13) (2007) 1643–1666.
[15] M.S. Akturk, A. Atamturk, S. Gurel, Parallel machine match-up scheduling with manufacturing cost considerations, J. Sched. 13 (1) (2010) 95–110.
[16] S. Gurel, E. Korpeoglu, M.S. Akturk, An anticipative scheduling approach with controllable processing times, Comput. Oper. Res 37 (6) (2010) 1002–1013.
[17] N. Al-Hinai, T.Y. ElMekkawy, Robust and stable flexible job shop scheduling with random machine breakdowns using a hybrid genetic algorithm, Int. J. Prod. Econ 132 (2) (2011) 279–291.
[18] T.C.E. Cheng, Q. Ding, B.M.T. Lin, A concise survey of scheduling with time-dependent processing times, Eur. J. Oper. Res. 152 (1) (2004) 1–13.
[19] V.S. Gordon, C.N. Potts, V.A. Strusevich, J.D. Whitehead, Single machine scheduling models with deterioration and learning: handling precedence constraints via priority generation, J. Sched 11 (5) (2008) 357–370.
[20] Y. Yin, T.C.E. Cheng, C.C. Wu, Scheduling with time-dependent processing times, Math. Prob. Eng 2014 (2014) 201421 Article ID.
[21] Y. Yin, S.-R. Cheng, J.Y. Chiang, J.C.H. Chen, X. Mao, C.C. Wu, Scheduling problems with due date assignment, Discrete Dyn. Nat. Soc 2015 (2015) 683269 Article ID.
[22] Y. Yin, W. Wu, T.C.E. Cheng, C.C. Wu, Due-date assignment and single-machine scheduling with generalised position-dependent deteriorating jobs and deteriorating multi-maintenance activities, Int. J. Prod. Res. 52 (8) (2014) 2311–2326.
[23] Y. Yin, M. Liu, J. Hao, M. Zhou, Single-machine scheduling with job-position-dependent learning and time-dependent deterioration, IEEE Trans. Syst. Man Cybern. Part A Syst. Humans 42 (1) (2012) 192–200.
[24] C.L. Zhao, H.Y. Tang, Scheduling deteriorating jobs under disruption, Int. J. Prod. Econ. 125 (2) (2010) 294–299.

[25] L. Liu, H. Zhou, Single-machine rescheduling with deterioration and learning effects against the maximum sequence disruption, Int. J. Syst. Sci. 46 (14) (2015) 2640–2658.
[26] D. Wang, F. Liu, J. Wang, Y. Wang, Integrated rescheduling and preventive maintenance for arrival of new jobs through evolutionary multi-objective optimization, Soft Comput. 2015, doi:10.1007/s00500-015-1615-7, in press.
[27] R. Graham, E. Lawler, J. Lenstra, K. Rinnooy, Optimization and approximation in deterministic sequencing and scheduling: a survey, Ann. Discrete Math 5 (1979) 287–326.
[28] M.A. Kubzin, V.A. Strusevich, Planning machine maintenance in two-machine shop scheduling, Oper. Res 54 (14) (2006) 789–800.
[29] Y. Jin, H. Branke, Evolutionary optimization in uncertain environments – a survey, IEEE Trans. Evol. Comput. 9 (3) (2005) 303–317.
[30] E.R.F.A. Schneider, R.A. Krohling, A hybrid approach using TOPSIS, Differential Evolution, and Tabu Search to find multiple solutions of constrained non-linear integer optimization problems, Knowl. Based Syst 62 (2014) 47–56.
[31] X. Shen, X. Yao, Mathematical modeling and multi-objective evolutionary algorithms applied to dynamic flexible job shop scheduling problems, Inf. Sci 298 (2015) 198–224.
[32] T. Duc-Hoc, M. Cheng, C. Minh-Tu, Hybrid multiple objective artificial bee colony with differential evolution for the time-cost-quality tradeoff problem, Knowl. Based Syst 74 (2015) 176–186.
[33] R. Zhang, S. Song, C. Wu, A two-stage hybrid particle swarm optimization algorithm for the stochastic job shop scheduling problem, Knowl. Based Syst 27 (2012) 393–406.
[34] I. Paenke, J. Branke, Y. Jin, Efficient search for robust solutions by means of evolutionary algorithms and fitness approximation, IEEE Trans. Evol. Comput. 10 (4) (2006) 405–420.
[35] J. Xiong, L.N. Xing, Y.W. Chen, Robust scheduling for multi-objective flexible job-shop problems with random machine breakdowns, Int. J. Prod. Econ. 141 (1) (2013) 112–126.
[36] D. Lim, Y. Jin, Y.S. Ong, B. Sendhoff, Generalizing surrogate-assisted evolutionary computation, IEEE Trans. Evol. Comput. 14 (3) (2010) 329–355.
[37] Y. Jin, Surrogate-assisted evolutionary computation: recent advances and future challenges, Swarm Evol. Comput 1 (2) (2011) 61–70.
[38] Y. Jin, M. Olhofer, B. Sendhoff, A framework for evolutionary optimization with approximate fitness functions, IEEE Trans. Evol. Comput. 6 (5) (2002) 481–494.
[39] B. Liu, Q. Zhang, G.G.E. Gielen, A Gaussian process surrogate model assisted evolutionary algorithm for medium scale expensive optimization problems, IEEE Trans. Evol. Comput 18 (2) (2014) 180–192.
[40] C. Sun, J. Zeng, J. Pan, S. Xue, Y. Jin, A new fitness estimation strategy for particle swarm optimization, Inf. Sci. 221 (2013) 355–370.
[41] A. Kattan, Y.S. Ong, Surrogate genetic programming: a semantic aware evolutionary search, Inf. Sci. 296 (2015) 345–359.
[42] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Trans. Evol. Comput. 6 (2) (2002) 182–197.
[43] C. Chang, C. Lin, LIBSVM, A library for support vector machines, ACM Trans. Intell. Syst. Technol. 2 (3) (2011) 1–27.
[44] A. Zhou, Q. Zhang, Y. Jin, Approximating the set of Pareto-optimal solutions in both the decision and objective spaces by an estimation of distribution algorithm, IEEE Trans. Evol. Comput 13 (5) (2009) 1167–1189.
[45] E. Zitzler, L. Thiele, et al., Multiobjective optimization using evolutionary algorithms – a comparative case study, in: A.E. Eiben, et al. (Eds.), Proceedings of the 5th International Conference on Parallel Problem Solving from Nature - PPSN V, Lecture Notes in Computer Science, 1498, Springe, Berlin, 1998, pp. 292–301.
[46] Q. Zhang, H. Li, MOEA/D: a multiobjective evolutionary algorithm based on decomposition, IEEE Trans. Evol. Comput. 11 (6) (2007) 712–731.
[47] H. Li, Q. Zhang, Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II, IEEE Trans. Evol. Comput. 13 (2) (2009) 284–302.
[48] J. Li, Q. Pan, K. Mao, P.N. Suganthan, Solving the steelmaking casting problem using an effective fruit fly optimisation algorithm, Knowl. Based Syst 72 (2014) 28–36.
[49] J. Shen, L. Wang, S. Wang, A bi-population EDA for solving the no-idle permutation flow-shop scheduling problem with the total tardiness criterion, Knowl. Based Syst 74 (2015) 167–175.
[50] S. Kiris, N. Yuzugullu, N. Ergun, A.A. Cevik, A knowledge-based scheduling system for emergency departments, Knowl. Based Syst 23 (2010) 890–900.
[51] X. Zheng, L. Wang, S. Wang, A novel fruit fly optimization algorithm for the semiconductor final testing scheduling problem, Knowl. Based Syst 57 (2014) 95–103.
[52] R. Cheng, Y. Jin, K. Narukawa, B. Sendhoff, A multiobjective evolutionary algorithm using Gaussian process based inverse modeling, IEEE Trans. Evol. Comput. (2015) in press, doi:10.1109/TEVC.2015.2395073.