A decomposition-based multi-objective particle swarm optimization algorithm with a local search strategy for key quality characteristic identification in production processes

Zhen He[a], Hao Hu[a], Min Zhang[a], An-Da Li[b,c,*]

[a] *College of Management and Economics, Tianjin University, Tianjin 300072, China*
[b] *School of Management, Tianjin University of Commerce, Tianjin 300134, China*
[c] *Research Center for Management Innovation and Evaluation, Tianjin University of Commerce, Tianjin 300134, China*

Abstract

Identifying the key quality characteristics (KQCs) (including part and process parameters) in production processes is essential for quality control. In this paper, we propose a data-driven KQC identification method based on production process data. We model KQC identification as a multi-objective feature selection problem of maximizing the geometric mean (GM) and minimizing the number of selected QCs (features). GM can evaluate the importance of a QC subset by measuring its predictive ability for product quality. To solve this optimization model, we propose a multi-objective optimization algorithm called MOPSO-LS that combines particle swarm optimization (PSO) with a local search strategy. MOPSO-LS adopts a decomposition approach, i.e., Tchebycheff approach (TA), to update personal best positions (*pbest*s) during the iterations. Thus, diversified and high quality solutions can be maintained by the *pbest*s of particles. Moreover, the local search strategy aims to update the non-dominated set found by MOPSO-LS during the iterations with two basic local search steps, i.e., a) adding and b) removing a feature, which can improve the convergence performance of MOPSO-LS. We have verified the proposed method on four production datasets. The experimental results indicate that MOPSO-LS can select a few KQCs with a good capability for predicting product quality, which shows the effectiveness of MOPSO-LS for KQC identification. Further comparisons show that MOPSO-LS obtains better search performance than four typical multi-objective optimization algorithms.

Keywords: Quality control, Feature selection, Particle swarm optimization, Decomposition approaches, Local search, Multi-objective optimization

1. Introduction

The production processes in modern industries generally contain a large number of part parameters (dimensional or performance features of parts) and process parameters, called quality characteristics (QCs),

*Corresponding author
Email address: adli@tjcu.edu.cn (An-Da Li)

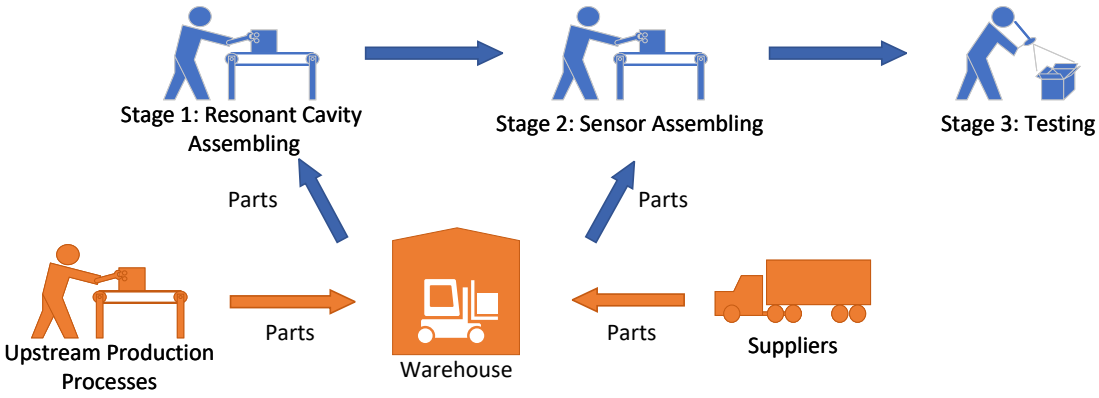

Figure 1: Illustration of the production process of the laser gyroscopes.

that determine the final quality level of products (Lee & Thornton, 1996). For example, Fig. 1 illustrates the production process of laser gyroscopes in one company. This production process contains two stages. At stage 1, the optical components are assembled in a cavity produced in the upstream production process. At stage 2, sensors are further assembled to the intermediate laser gyroscope produced at stage 1 to produce the final product. After the two stages, the performance of laser gyroscopes is tested and different quality levels (e.g., conforming or nonconforming) are assigned to the produced laser gyroscopes. The part and process parameters (i.e., QCs) at each stage are the potential factors that decrease the quality of the final products. It is required to select the key QCs (KQCs) that significantly affect product quality, which is termed the KQC identification problem. The identified KQCs can be used to build an effective quality prediction model for advanced quality control (Chien et al., 2022). In the Internet of Things (IoT) era, the production data can be sufficiently collected from the production processes with the wide application of sensor devices (Yang et al., 2019; Kang et al., 2020). It is worth proposing a data-driven KQC identification method using the high dimensional production data with a huge number of QCs.

The data-driven KQC identification methods generally determine the KQCs strongly related to product quality by analyzing the correlations between QCs (input variables) and the product/process quality (response variable). Many studies have been proposed to identify KQCs based on linear regression models. For example, Jin & Zhou (2006) proposed an eigenspace comparison method built with the linear regression model to identify the causes of variations that reduce product quality. Li & Chen (2016) proposed a Bayesian approach to identify both the process mean and sensor faults in the production process that lead to defective products. Wang & Shi (2021) proposed a holistic approach to model and analyze the multistage manufacturing process, where an ℓ_2 penalty was added in the linear regression model to select the key input variables strongly related to the product quality. These approaches can simultaneously select the KQCs and build a regression model that depicts the relationship between KQCs and the product/process quality.

However, these approaches do not work well for the production processes with a very large number of QCs and non-linear variable correlations.

Feature selection (FS) (Guyon & Elisseeff, 2003; Xue et al., 2016) in machine learning aims at selecting the key features related to the class label in order to build an effective learning algorithm. Due to the effectiveness for high dimensional data, FS methods (Jeong & Cho, 2006; Anzanello et al., 2009, 2012; Li et al., 2016; Chien et al., 2017; Guo & Banerjee, 2017; Li et al., 2020; Wang et al., 2020) have been extensively applied to select the KQCs (features) in production processes that have excellent predictive performance for product quality in recent years. FS can be categorized into the filter and wrapper approaches according to the principle used for evaluating the feature importance. A filter method uses a measure based on the distance (Robnik-Sikonja & Kononenko, 2003), information theory (Yu & Liu, 2004), statistical theory (Sugiyama, 2007), etc., to evaluate the importance of features. For example, Chien et al. (2017) studied the root cause detection problem for the semiconductor production process. In this study, the Cramer's V correlation coefficient is used to group the correlated process parameters, and then, the Akaike information criterion is utilized to select the key process parameters (i.e., root causes) in each group that affect the yield of semiconductors. Wang et al. (2020) proposed an FS method to identify key factors that affect the cycle time of the semiconductor wafer fabrication process. A network deconvolution method was combined with the mutual information measure to identify key features. Different from filter methods, a wrapper method (Kohavi & John, 1997) evaluates the feature importance by involving a learning algorithm. The predictive performance of the learning algorithm is used to evaluate the importance of different feature subsets. For example, Li et al. (2016) proposed a wrapper method that adopts a Naive Bayesian (NB) classifier to evaluate the importance of QCs and adopts non-dominated sorting genetic algorithm II (NSGA-II) to search for the optimal QC subset, i.e., KQC set. The features selected by a wrapper method can generally build a learning algorithm with better predictive performance than those selected by a filter method since the learning algorithm is directly used in a wrapper method.

The FS method is generally defined as an optimization problem that aims to find a feature subset with a small size and excellent classification performance for the class label. This optimization problem is proven to be an NP-hard problem that has $2^D - 1$ feasible solutions (D is the number of original features) (Amaldi & Kann, 1998; Li et al., 2021). To solve this challenging optimization problem, heuristic search algorithms, such as sequential forward selection (SFS) (Kohavi & John, 1997) and sequential backward selection (SBS) (Kohavi & John, 1997), were adopted in the literature. The evolutionary computation (EC) techniques such as genetic algorithms (GAs) (Oh et al., 2004; Li et al., 2020; Liu et al., 2022), particle swarm optimization (PSO) (Li et al., 2021; Nguyen et al., 2021; Zhang et al., 2022), differential evolution (DE) (Guan et al., 2021; Wang et al., 2021), and ant colony optimization (ACO) (Wan et al., 2016) have also been used for FS due to their excellent search abilities. Specifically, the PSO algorithms are increasingly applied to FS for their good search performance and fast convergence speed (Xue et al., 2016).

The multi-objective EC techniques (Nguyen et al., 2020; Aljarah et al., 2020; Zhang et al., 2020; Kiziloz & Deniz, 2021) have also been applied to FS because two conflicting objectives, i.e., maximizing the classification performance and minimizing the number of selected features, are involved in an FS problem. Specifically, accuracy is generally utilized to evaluate the classification performance of feature subsets in most multi-objective EC based FS methods. However, accuracy may not be a good classification performance measure for unbalanced data because its value is mainly affected by the majority class instances in the data. This can affect the final FS performance. To cope with this data imbalance problem, other classification performance measures are adopted instead of accuracy in the FS methods. For example, Pacheco et al. (2013) proposed a multi-objective FS method that optimizes the type I and type II errors using NSGA-II. Kozodoi et al. (2019) adopted NSGA-II to solve an FS model that is defined as maximizing an expected maximum profit measure (a cost-sensitive classification performance measure considering data imbalance) and minimizing the number of selected features. Li et al. (2019) proposed an improved direct multi-search (IDMS) method to optimize an FS model that is defined as maximizing the geometric mean (GM) (i.e., geometric mean of sensitivity and specificity) and minimizing the number of selected features. This IDMS method was applied to select the KQCs in production processes based on unbalanced production data.

In this paper, we aim to build a data-driven method to identify the KQCs in the production process that have good predictive performance for the quality of final products. We will build a wrapper-based FS method for KQC identification because the nature of evaluating the predictive performance of QCs during the FS process makes wrappers have good FS performance. Specifically, KQC identification is modeled as a multi-objective FS problem of maximizing the GM and minimizing the number of selected QCs (features) (Li et al., 2019), which can address the unbalanced production data. The high dimensionality of data creates a huge search space for the modeled FS problem. So, standard multi-objective EC techniques (without using some strategies to accelerate the convergence speed) would require substantial computational resources to identify the KQCs by searching in the huge space. Therefore, we propose a multi-objective PSO algorithm called MOPSO-LS that combines the mechanisms of PSO and the local search for the multi-objective FS problem. The local search is applied to improve the convergence speed and search performance of PSO. The contributions of the proposed MOPSO-LS algorithm are listed below:

- We propose a decomposition-based *pbest* (personal best position) update (DPU) strategy for MOPSO-LS. The DPU strategy applies the Tchebycheff approach (TA), a decomposition approach, to obtain the aggregated objective function (based on the original multiple objective functions) for each particle in the swarm. The aggregated objective function value determines the update of each *pbest* in the swarm. Moreover, similar to MOEA/D (multi-objective evolutionary algorithm based on decomposition), we use the new position of a particle to update the *pbest*s of a set of the neighbor particles instead of only updating its own *pbest*. This means that a particle can also learn from the best

experience of its neighbor particles, which can improve the convergence speed of the algorithm. It should be noted that the ranges of different objective functions are very different, which can result in the incommensurability problem when adopting the TA. In DPU, we modify the original TA by introducing one additional reference point. Thus, the modified TA can normalize different objective function values before obtaining the aggregated function value.

- We propose a local search step for MOPSO-LS. The local search step conducts two basic operations, adding and removing, to update each non-dominated solution at each iteration after the particle swarm based optimization step. The adding operation randomly adds a feature to the current solution (feature/QC subset), while the removing operation randomly eliminates a feature from the current solution. Thus, for each non-dominated solution, two new solutions are generated by the local search step. The new solutions of the local search step are used to further update the non-dominated set. The local search step aims to purify the non-dominated set of MOPSO-LS at each iteration, and thus it can improve the convergence performance of MOPSO-LS.

- We have verified the KQC identification performance of MOPSO-LS on four unbalanced production datasets in the experiments. The results show that the proposed MOPSO-LS can obtain a few KQCs that have good predictive performance for the quality level of products, which indicates good KQC identification performance. Further analysis indicates that MOPSO-LS obtains better search performance than four typical multi-objective optimization algorithms.

The remainder of this paper is organized as follows. Section 2 presents the preliminaries of the proposed method, including the descriptions of PSO and TA. Section 3 describes the KQC identification problem addressed in this paper. Section 4 describes the details of the proposed optimization algorithm MOPSO-LS. Section 5 gives the experimental design. Section 6 presents the KQC identification results of the experiments and gives the discussion. Section 7 further analyzes the search performance of MOPSO-LS. Finally, the conclusions and future research interests are given in Section 8.

2. Preliminaries

2.1. Particle swarm optimization (PSO)

PSO (Kennedy & Eberhart, 1995) is a popular swarm-based optimization approach that has been widely used in many real-world optimization problems. It performs the optimization process using a swarm of particles. The particles continuously change their positions (solutions) based on their *pbest*s and the global best position (*gbest*) to search for the optimal solution.

Considering the optimization problem with a feasible solution region $\Omega \subseteq \mathbb{R}^D$, we use a D-dimensional vector $\mathbf{x} \in \Omega$ to represent a feasible solution. Two vectors, a position vector and a velocity vector, depict

the status of a particle in the swarm in PSO. The position vector shows the current position of the particle and the velocity vector describes the moving tendency of the particle. Therefore, the particle positions can be updated during the iterations of PSO. Let the position vector and velocity vector for particle i at the tth iteration be $\mathbf{x}_i^t = (x_{i,1}^t, x_{i,2}^t, ..., x_{1,D}^t)^T$ and $\mathbf{v}_i^t = (v_{i,1}^t, v_{i,2}^t, ..., v_{1,D}^t)^T$. The velocity $\mathbf{v}_i^t$ is updated as

$$\mathbf{v}_i^{t+1} = w\mathbf{v}_i^t + c_1\mathbf{r}_i^1 \odot (\mathbf{p}_i - \mathbf{x}_i^t) + c_2\mathbf{r}_i^2 \odot (\mathbf{g} - \mathbf{x}_i^t) \tag{1}$$

where $\mathbf{r}_i^1 = (r_{i,1}^1, r_{i,2}^1, ..., r_{i,D}^1)^T$ and $\mathbf{r}_i^2 = (r_{i,1}^2, r_{i,2}^2, ..., r_{i,D}^2)^T$ are two random vectors in which $r_{i,d}^1$ and $r_{i,d}^2$ $(d = 1, 2, ..., D)$ follow the uniform distribution $U(0, 1)$. $\mathbf{p}_i$ and $\mathbf{g}$ are two D-dimensional vectors that represent the *pbest* previously reached by the particle i and the *gbest* previously reached by the swarm. $\odot$ is the Hadamard product that performs the element-wise product operation based on two vectors with the same shape to create a new vector. w is a predefined inertia weight that controls the impact of the previous velocity on the new velocity, and c_1 and c_2 are two predefined accelerate parameters that reflect the impact of *pbest* and *gbest* on updating the velocity. With the updated velocity $\mathbf{v}_i^{t+1}$, the particle position $\mathbf{x}_i^t$ is updated to

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \mathbf{v}_i^{t+1}. \tag{2}$$

The velocities and positions of the particles in the swarm are updated with Eqs. (1) and (2) iteratively during the evolutionary process of PSO. The *gbest* is finally output as the found optimal solution when the termination condition reaches.

The FS methods based on multi-objective PSO algorithms have been extensively studied in recent years due to the good search performance of PSO (Xue et al., 2013; Nguyen et al., 2016; Amoozegar & Minaei-Bidgoli, 2018; Zhou et al., 2021; Han et al., 2021). For example, Xue et al. (2013) applied two multi-objective PSO algorithms called CMDPSO (PSO based on crowding, mutation, and dominance) and NSPSO (non-dominated sorting PSO) to FS. CMDPSO adopts the idea of mutation and crowding distance to improve the swarm diversity during the optimization process. NSPSO adopts the idea of non-dominated sorting in NSGA-II for maintaining good particles during the iterations. Zhou et al. (2021) proposed a flexible cut-point PSO (FCPSO) for FS and feature discretization. The proposed FCPSO combines the solution updating strategy of bare-bone PSO with a mutation operator to improve the search performance. Moreover, the local search step has also been used in multi-objective PSO algorithms to improve the FS performance. For example, Nguyen et al. (2016) proposed a multi-objective PSO algorithm called ISRPSO (PSO with inserting, swapping, and removing operations) for FS. This algorithm adopts three basic local search operations, i.e., insert, remove, and swap, to update the archive (non-dominated set) at each iteration after the PSO-based solution updating step. However, the local search operations are conducted in a sequential strategy similar to SFS and SBS, which yields a large number of local search steps at an iteration. Thus, the computational time

of the local search step and the whole algorithm is substantially improved. Amoozegar & Minaei-Bidgoli (2018) proposed a multi-objective PSO algorithm called RFPSOFS (ranked feature PSO feature selection) that applied a local search step to refine the archive. The local search step ranks the features based on their selection times by the particles and divides the features into different subgroups. A new solution is generated by selecting a predefined number of features from different subgroups. However, it is not easy to properly define the number of subgroups and the number of selected features from the subgroups.

The *pbest* and *gbest* are two key components that affect the search behavior of PSO. In the multi-objective scenario, the *gbest* can be selected from the currently found non-dominated set (archive) based on the definition of *gbest* in PSO. The *pbest* is generally determined based on the Pareto dominance concept (Xue et al., 2013; Nguyen et al., 2020), i.e., the *pbest* is updated if a new particle position dominates the current *pbest*. However, this dominance-based update strategy may not be a good way to update *pbest* for a multi-objective PSO algorithm. This is because there is a high probability that the new position and *pbest* do not dominate each other, which significantly reduces the chances of updating *pbest* and further affects the search performance of PSO. In recent years, the decomposition approaches have been used in many EC techniques, including multi-objective genetic local search (MOGLS) (Ishibuchi & Murata, 1998), MOEA/D (Zhang & Li, 2007), and decomposition-based interactive evolutionary algorithm (Tomczyk & Kadziński, 2020), to address multi-objective optimization problems (MOPs). The decomposition approaches can be used to update *pbest*. They can convert the multiple objectives to be optimized into an aggregated objective, which can be used to determine the update of *pbest*. With a decomposition approach, any two solutions (even if they do not dominate each other) can be compared. Therefore, a decomposition approach can be a better way of updating *pbest* than the dominance-based update strategy. Moreover, the decomposition approaches can well maintain the diversity of solutions with the *pbest*s of particles because we can assign different weight vectors to different particles to form their own aggregated objectives (Han et al., 2021). Thus, the search performance of the algorithm can be improved.

2.2. Decomposition of multi-objective optimization: Tchebycheff approach (TA)

The decomposition approaches aim to convert the task of approximating the Pareto front (PF) of a MOP into a number of single objective optimization problems. The general used decomposition approaches include the weighted sum approach (WSA), the TA, and the penalty boundary intersection approach (PBIA) (Zhang & Li, 2007). These three approaches have different characteristics for solving MOPs: a) WSA can work well when the PF is convex, but it is weak in finding non-convex PFs; b) TA can address the weakness of WSA in obtaining non-convex PFs, however, the converted objective function in TA is not smooth (non-differentiable) which proposes challenges for derivative-based optimization algorithms; c) PBIA has the advantage over TA that it can find more uniformly distributed solutions when there are more than two objectives, however, it needs a properly predefined penalty parameter θ to ensure its performance. In this

paper, the TA is utilized to build a decomposition-based multi-objective PSO algorithm (i.e., the MOPSO-LS) for KQC identification. Because the multi-objective KQC identification model to be optimized has two objectives and PSO is a deviation-free optimization algorithm, the above-mentioned disadvantage of TA can be avoided.

Without loss of generality, suppose that we are considering a MOP with m objectives to be minimized:

$$\begin{aligned} &\text{minimize} \quad \mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), ..., f_m(\mathbf{x}))^T \\ &\text{subject to} \quad \mathbf{x} \in \Omega \end{aligned} \tag{3}$$

where $\mathbf{x}$ is a solution, $\Omega \subseteq \mathbb{R}^D$ is the feasible solution region, and $f_j(\mathbf{x})$ $(j = 1, 2, ..., m)$ is the jth objective function. Then, we can use TA to convert the MOP in Eq. (3) into the following K single objective models:

$$\begin{aligned} &\text{minimize} \quad g(\mathbf{x}|\boldsymbol{\lambda}^k, \mathbf{z}^*) = \max_{j \in \{1,2,...,m\}} \{\lambda_j^k | f_j(\mathbf{x}) - z_j^*|\} \\ &\text{subject to} \quad \mathbf{x} \in \Omega \end{aligned} , k = 1, 2, ..., K, \tag{4}$$

where $\mathbf{z}^* = (z_1^*, ..., z_m^*)^T$ is the reference point defined in the objective space, and each z_j^* $(j = 1, 2, ..., m)$ is defined as $z_j^* = \min_{\mathbf{x} \in \Omega} f_j(\mathbf{x})$. $\boldsymbol{\lambda}^k = (\lambda_1^k, ..., \lambda_m^k)^T$ is a weight vector where each λ_j^k $(j = 1, 2, ..., m)$ denotes the weight of the jth objective function for constructing the aggregated objective function $g(\mathbf{x}|\boldsymbol{\lambda}^k, \mathbf{z}^*)$. Defining $\boldsymbol{\lambda}^1, \boldsymbol{\lambda}^2, ..., \boldsymbol{\lambda}^K$ as K evenly spread vectors, a set of evenly distributed solutions that approximate the true PF can be found by solving the converted K single objective optimization models defined in Eq. (4).

3. Problem description

3.1. Modeling the production process

A multistage production process with L stages is illustrated in Fig. 2, where $\mathbf{u}_i = (u_{i,1}, u_{i,1}, ..., u_{i,J_i})^T$ $(i = 1, 2, ..., L)$ is a vector that denotes the QCs (part and process parameters) at stage i ($u_{i,j}$, $j = 1, 2, \ldots, J_i$ denotes the jth QC), and $\mathbf{y}_i$ is a vector of quality variables for the intermediate $(i = 1, 2, ..., L-1)$ or final $(i = L)$ product produced at stage i (Shi & Zhou, 2009). Following the assumption in Tsung et al. (2008), we assume that the quality variables $\mathbf{y}_i$ $(i = 1, 2, \ldots, L-1)$ of the intermediate products are unobserved. For the ith stage, the quality variables of the previous stage (i.e., $\mathbf{y}_{i-1}$) and the QCs $\mathbf{u}_i$ are the factors that affect the quality $\mathbf{y}_j$ of the intermediate/final product. The relationship between $\mathbf{y}_{i-1}$, $\mathbf{u}_i$, and $\mathbf{y}_i$ can be modeled as

$$\mathbf{y}_i = \begin{cases} \mathcal{F}_i(\mathbf{u}_i), & i = 1 \\ \mathcal{F}_i(\mathbf{u}_i, \mathbf{y}_{i-1}), & \forall i \in \{2, 3, ..., L\} \end{cases}. \tag{5}$$

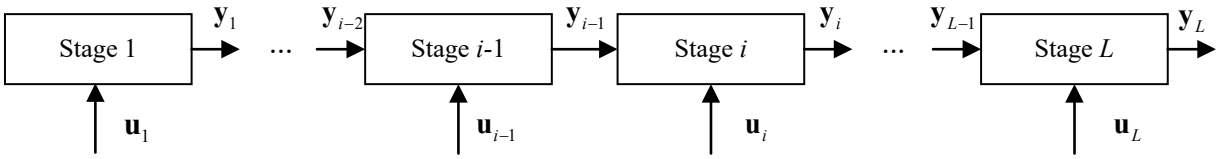

Figure 2: Illustration of a production process with L stages.

Since the input $\mathbf{y}_{i-1}$ at stage i is the output of the previous stage $i-1$, we can aggregate the separate relation models shown in Eq.(5) and obtain the following model:

$$\mathbf{y}_L = \mathcal{F}_{ag1}(\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_L). \tag{6}$$

According to Eq. (6), the QCs at each stage, i.e., $\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_L$, are the factors that affect the quality of final products. In this paper, we address the production process where the quality of final products is measured by one variable (e.g., conforming or nonconforming). Thus, we can replace the vector of quality variables $\mathbf{y}_L$ with a scalar variable Y, which denotes the quality of final products. Moreover, supposing that the total number of QCs of all the L stages is D, the QCs at all stages can be denoted by a new vector $\mathbf{Q} = (Q_1, Q_2, ..., Q_D)^T$. So the relation model in Eq. (6) can be rewritten as

$$Y = \mathcal{F}_{ag2}(\mathbf{Q}) \tag{7}$$

3.2. KQC identification model

According to Eq. (7), the QCs, i.e., $Q_1, Q_2, ..., Q_D$, are the factors that affect the quality Y of final products. However, from a practical point of view, not all these QCs are equally important in terms of product quality. Some QCs may be well controlled and the quality of final products is not sensitive to variations of these QCs, and variations of some other QCs may significantly deteriorate the product quality. It is required to select the KQCs strongly related to product quality to effectively control the quality of products. Let $\mathbb{Q} = \{Q_1, Q_2, ..., Q_D\}$ be the set of original QCs. We define KQC identification as an FS problem of selecting a QC (feature) subset $\mathbb{X} \subseteq \mathbb{Q}$ that has competent predictive performance for product quality (the class label). In this paper, we address the production process where the products have two quality levels. A mathematical description of this KQC identification problem is introduced as follows.

Let $\mathcal{D} = \{\boldsymbol{\mathcal{Q}}, \mathbf{y}\}$ be the production dataset with M products (instances) and D QCs, where $\boldsymbol{\mathcal{Q}} \in \mathbb{R}^{M \times D}$ is a matrix denoting the measurements the QCs (features), i.e., $Q_1, Q_2, ..., Q_D$, and $\mathbf{y} \in \mathbb{B}^{M \times 1}$ ($\mathbb{B} \in \{-1, +1\}$) is a vector denoting the measured quality level Y (i.e., the class label) for the products. Generally, two possible scenarios would appear in a dataset collected from a production process: a) most products are

conforming and a few products are nonconforming, and b) most products are regular and a few products are premium. Both these two scenarios create an imbalance issue for the production dataset $\mathcal{D}$. In this paper, $Y = +1$ is the minority/positive class which denotes the minority quality level (nonconforming/premium), and $Y = -1$ is the majority/negative class which denotes the majority quality level (conforming/regular).

With the dataset $\mathcal{D}$, we can establish an empirical model that depicts the relationship between QCs and product quality. In this paper, the NB classifier (John & Langley, 1995) is used to build the empirical model for its high performance and conciseness. Then, based on the wrapper framework, KQC identification (FS) can be formulated as an optimization problem of maximizing the classification performance (for product quality) of the NB classifier by selecting the optimal QC subset $\mathbb{X}^* \subseteq \mathbb{Q}$, and we obtain the following KQC identification model:

$$\begin{aligned} &\text{minimize} \quad \mathbf{F}(\mathbb{X}) = (1 - p(\mathbb{X}), |\mathbb{X}|)^T \\ &\text{subject to} \quad \mathbb{X} \subseteq \mathbb{Q} \end{aligned} \tag{8}$$

where $\mathbb{X}$ is a QC/feature subset, $p(\mathbb{X})$ denotes the classification performance of the NB classifier for the quality level of products by using $\mathbb{X}$ as the input QCs, and $|\mathbb{X}|$ denotes the number of QCs in $\mathbb{X}$. In Eq.(8), minimizing $|\mathbb{X}|$ is adopted as the second objective so the model's ability to eliminate the irrelevant and redundant QCs can be improved. It should be noted that both the continuous and discrete QCs can be handled by the KQC identification model defined in Eq.(8) because the NB classifier (John & Langley, 1995) can address both the continuous and discrete features.

The classification performance measure $p(\mathbb{X})$ in Eq.(8) should be further defined for the KQC identification model. The confusion matrix in binary classification is shown in Table 1. The TP/TN indicates the number of actual positive/negative instances that are correctly classified into the positive/negative class. The FP/FN indicates the number of actual negative/positive instances that are incorrectly classified into the positive/negative class. Based on the notations in Table 1, we can define the classification performance measure GM as

Table 1: Confusion matrix.

Predicted condition / Actual condition	Positive (+1)	Negative (-1)
Positive (+1)	True positive (TP)	False negative (FN)
Negative (-1)	False positive (FP)	True negative (TN)

$$GM = \sqrt{Sensitivity * Specificity} \tag{9}$$

where

$$Sensitivity = \frac{TP}{TP + FN} \tag{10}$$

and

$$Specificity = \frac{TN}{TN + FP}. \tag{11}$$

According to Eqs. (10) and (11), sensitivity is defined as the proportion of correctly classified positive instances to all actual positive instances, and specificity is defined as the proportion of correctly classified negative instances to all actual negative instances. Thus, the geometric mean (i.e., GM) of sensitivity and specificity is sensitive to the classification performance of both positive and negative instances, which makes GM a good measure for unbalanced production data. The GM measure is used as the classification performance measure in Eq. (8) (i.e., $p(\mathbb{X}) = GM(\mathbb{X})$) to build a multi-objective KQC identification model in our method to address the unbalanced production data.

4. Proposed multi-objective optimization algorithm: MOPSO-LS

We propose a multi-objective optimization algorithm called MOPSO-LS to solve the multi-objective KQC identification model defined in Eq. (8). MOPSO-LS combines the particle swarm based optimization mechanism with a local search mechanism. To adapt to MOPs, the decomposition approach TA is utilized to construct MOPSO-LS. Because MOPSO-LS finds a set (denoted by Θ) of non-dominated solutions (QC subsets), it is required to select the best compromise solution from Θ as the KQC set. In this paper, the ideal point method (IPM) is adopted to select the best compromise solution. The procedure of IPM can be found in Li et al. (2016). In the following subsections, the details of the proposed MOPSO-LS algorithm will be introduced.

4.1. Overall procedure

The overall procedure of MOPSO-LS is shown in Algorithm 1. Since we aim to solve the FS (KQC identification) problem defined in Eq. (8), a dataset is required. We divide the original dataset $\mathcal{D}$ into a training set $\mathcal{D}_{tr}$ and a test set $\mathcal{D}_{te}$. $\mathcal{D}_{tr}$ is input to the algorithm for evaluating the objective function values based on the KQC identification model. $\mathcal{D}_{te}$ is the unseen data used to verify the effectiveness of the identified KQCs.

As shown in Algorithm 1, the overall procedure of MOPSO-LS includes the "1. Initialization" and "2. Iterations" steps. In the *Initialization* step, a swarm $\mathbb{S}^t$ $(t = 0)$ of N particles is initialized. The solutions (particle positions) in $\mathbb{S}^t$ are used to update the non-dominated set Θ and the *pbest* (denoted by $\mathbf{p}_i$) of each particle $i \in \{1, 2, ..., N\}$.

In the *Iterations* step, we adopt two search steps, including "2.1 Particle swarm based optimization" and "2.2 Local search", to update the solutions at each iteration. First, in the *particle swarm based optimization* step, the following operations are conducted:

Algorithm 1: Pseudocode of the MOPSO-LS algorithm.

Input : The training set $\mathcal{D}_{tr}$, the swarm size N, a set of N uniformly spread vectors $\{\boldsymbol{\lambda}^1, \boldsymbol{\lambda}^2, ..., \boldsymbol{\lambda}^N\}$, the size of neighbors T ;

Output : The non-dominated set Θ ;

`/* 1. Initialization */`

1 $t \leftarrow 0$; `/* Initialize the iteration counter. */`

2 $\Theta \leftarrow \emptyset$; `/* Initialize the non-dominated set. */`

3 For each particle $i = 1, 2, ..., N$, set its neighbor index set $\mathbb{B}_i = \{i_1, ..., i_T\}$, where $\boldsymbol{\lambda}^{i_1}, \boldsymbol{\lambda}^{i_2}, ..., \boldsymbol{\lambda}^{i_T}$ are the T closest vectors of $\boldsymbol{\lambda}^i$;

4 $\mathbb{S}^t \leftarrow \{(\mathbf{x}_1^t, \mathbf{v}_1^t), (\mathbf{x}_2^t, \mathbf{v}_2^t), ..., (\mathbf{x}_N^t, \mathbf{v}_N^t)\}$; `/* Initialize a swarm of` N `particles, where` $\mathbf{x}_i^t$ `and` $\mathbf{v}_i^t$ `denote the position and velocity of the` i`th (` $i = 1, 2, ..., N$`) particle. */`

5 Evaluate the objective function values $(f_1(\mathbf{x}), f_2(\mathbf{x}))^T$ using Eq. (8) for each particle position $\mathbf{x}$ in $\mathbb{S}^t$;

6 For each particle $i \in \{1, 2, ..., N\}$, set its *pbest* as $\mathbf{p}_i = \mathbf{x}_i^t$;

7 $\Theta \leftarrow$ Update non-dominated set Θ with the particle positions in $\mathbb{S}^t$;

`/* 2. Iterations */`

8 **repeat**

`/* 2.1 Particle swarm based optimization */`

9 $\mathbf{g} \leftarrow$ Select *gbest* from Θ ;

10 **for** *particle* $i \leftarrow 1, 2, ..., N$ **do**

11 $\mathbf{v}_i^{t+1} \leftarrow$ Update the velocity vector based on $\mathbf{g}$, $\mathbf{p}_i$, and $\mathbf{x}_i^t$ using Eq. (1);

12 $\mathbf{x}_i^{t+1} \leftarrow$ Update the position vector based on $\mathbf{x}_i^t$ and $\mathbf{v}_i^{t+1}$ using Eq. (2);

13 $\mathbf{x}_i^{t+1} \leftarrow$ Update $\mathbf{x}_i^{t+1}$ with the mutation operation (see Section 4.3).

14 **end**

15 $\mathbb{S}^{t+1} \leftarrow \{(\mathbf{x}_1^{t+1}, \mathbf{v}_1^{t+1}), (\mathbf{x}_2^{t+1}, \mathbf{v}_2^{t+1}), ..., (\mathbf{x}_N^{t+1}, \mathbf{v}_N^{t+1})\}$;

16 Evaluate the objective function values using Eq. (8) for each particle in $\mathbb{S}^{t+1}$;

`/* Update the reference points. */`

17 $\mathbf{z}^* = (z_1^*, z_2^*)^T \leftarrow (\min_{\mathbf{x} \in \{\mathbb{S}^{t+1} \cup \Theta\}} f_1(\mathbf{x}), \min_{\mathbf{x} \in \{\mathbb{S}^{t+1} \cup \Theta\}} f_2(\mathbf{x}))^T$, $\mathbf{z}' = (z_1', z_2')^T \leftarrow (\max_{\mathbf{x} \in \Theta} f_1(\mathbf{x}), \max_{\mathbf{x} \in \Theta} f_2(\mathbf{x}))^T$;

18 Update the *pbest* $\mathbf{p}_i$ for each particle $i \in \{1, 2, ..., N\}$ using the proposed decomposition-based *pbest* update (DPU) strategy (see Section 4.4) ;

19 $\Theta \leftarrow$ Update the non-dominated set Θ with particle positions in $\mathbb{S}^{t+1}$;

`/* 2.2 Local search */`

20 $\mathbb{L} \leftarrow$ Perform the local search (see Section 4.5) based on the solutions in Θ ;

21 $\Theta \leftarrow$ Update the non-dominated set Θ with the solutions in $\mathbb{L}$;

22 $t \leftarrow t + 1$;

23 **until** *the termination condition*;

24 **return** The non-dominated set Θ ;

a) In the multi-objective optimization scenario, more than one best solution is generally stored in the non-dominated set Θ during the iterations. So, different from single objective PSO that has only one certain *gbest* at an iteration, any solution in Θ can be a candidate for *gbest* in MOPSO-LS. Therefore, we need to select a *gbest* vector $\mathbf{g}$ from Θ at each iteration (as shown in line 9 of Algorithm 1). To make each solution in Θ be uniformly selected, we use a counter variable c_g for each solution in Θ to record the number of times this solution is selected as the *gbest*. During the selection of *gbest*, the solution with the smallest c_g value is selected (if several solutions in Θ have the same smallest c_g value, we select *gbest* from these solutions at random). This selection strategy for *gbest* is beneficial for the algorithm to uniformly allocate computational resources to different regions of Θ.

b) As shown in lines 10 to 14 of Algorithm 1, the new velocity $\mathbf{v}_i^{t+1}$ and position $\mathbf{x}_i^{t+1}$ for each particle i is first obtained with the PSO-based solution update mechanism shown in Eqs. (1) and (2). After that, we adopt a mutation operation to further update the new position $\mathbf{x}_i^{t+1}$ to avoid the premature convergence problem. Then, the swarm $\mathbb{S}^{t+1}$ for the next iteration is obtained and the objective function values for each particles are evaluated based on the optimization model in Eq. (8).

c) As shown in line 17 of of Algorithm 1, two reference points $\mathbf{z}^*$ and $\mathbf{z}'$ are updated based on the objective function values $(f_1, f_2)^T$ of solutions in $\mathbb{S}^{t+1}$ and the non-dominated set Θ. Specifically, $\mathbf{z}^*$ is updated as the minimum objective function values found by the algorithm so far, and $\mathbf{z}'$ is updated as the maximum objective function values obtained by current Θ. Given the two reference points and the new particle positions in $\mathbb{S}^{t+1}$, the *pbset* $\mathbf{p}_i$ for each particle i $(i = 1, 2, ..., N)$ is updated using the proposed DPU strategy. The non-dominated set Θ is updated as well with $\mathbb{S}^{t+1}$.

After the *particle swarm based optimization*, the *local search* step (as shown in line 20 of Algorithm 1) is conducted to locally update each solution in Θ. Θ is updated with the new solutions generated by the local search step. Finally, the algorithm output Θ when the termination condition reaches.

In the following subsections, the main components of MOPSO-LS, including solution encoding, mutation operation, the DPU strategy, and local search, are introduced.

4.2. Solution encoding

In the proposed MOPSO-LS algorithm, the position of each particle i is designed as a D-dimensional real vector $\mathbf{x}_i^t = (x_{i,1}^t, x_{i,2}^t, ..., x_{1,D}^t)^T$, where each $x_{i,d}^t \in [0, 1]$ $(d = 1, 2, ..., D)$. This position vector represents a solution $\mathbb{X}$ (QC/feature subset) for the KQC identification problem of D original features given a threshold parameter τ. Specifically, the dth QC is selected by $\mathbf{x}_i^t$ if $x_{i,d}^t > \tau$ and is eliminated by $\mathbf{x}_i^t$ if $x_{i,d}^t \leq \tau$. The velocity $\mathbf{v}_i^t = (v_{i,1}^t, v_{i,2}^t, ..., v_{1,D}^t)^T$ for each particle is a real vector where each $v_{i,d}^t \in [-v_{max}, v_{max}]$ ($d = 1, 2, ..., D$). In this paper, we set $\tau = 0.6$ and $v_{max} = 0.6$ as suggested in Xue et al. (2013). During the initialization step of MOPSO-LS, each element in the position $\mathbf{x}_i^t$ is randomly initialized in $[0, 1]$ and each element in the velocity $\mathbf{v}_i^t$ is initialized as 0. Note that, during the iterations of MOPSO-LS, if any $x_{i,d}^t$ in a new position $\mathbf{x}_i^t$ is out of the range $[0, 1]$, we replace its value with $x_{i,d}^t = 0$ or $x_{i,d}^t = 1$ if $x_{i,d}^t < 0$ or $x_{i,d}^t > 1$. Similarly, if any $v_{i,d}^t$ in a new velocity $\mathbf{v}_i^t$ is out of the range $[-0.6, 0.6]$, we replace its value with $x_{i,d}^t = -0.6$ if $x_{i,d}^t < -0.6$, and replace its value with $x_{i,d}^t = 0.6$ if $x_{i,d}^t > 0.6$.

4.3. Mutation operation

In GAs, the mutation operation is used to avoid the premature convergence problem. Similarly, in MOPSO-LS (as shown in line 13 of Algorithm 1), we propose a mutation operation to further update the position of each particle after the solution updating mechanism of standard PSO shown in Eqs. (1) and (2) to improve its search performance. This mutation operation is conducted on each variable $x_{i,d}^t$ in a particle

position $\mathbf{x}_i^t = (x_{i,1}^t, x_{i,2}^t, ..., x_{1,D}^t)^T$ with a probability of p_m. A variable $x_{i,d}^t \leq \tau$ is changed to a random value in $(\tau, 1]$ and a variable $x_{i,d}^t > \tau$ is changed to a random value in $[0, \tau]$. Thus, the mutation operation changes the state that whether the dth QC is selected or not.

4.4. Decomposition-based pbest update (DPU) strategy

In the single objective optimization scenario, the *pbest* of each particle is updated when the objective function value of a new position is better than that of the current *pbest*. This strategy is inherited in the two multi-objective PSO algorithms CMDPSO and NSPSO (Xue et al., 2013), where *pbest* is updated when the new position of the particle dominates (decided using the Pareto dominance concept) the current *pbest*. However, this strategy may not work well for updating *pbest* in a multi-objective PSO algorithm. The update of *pbest* is difficult to be reached with the dominance-based strategy because there is a relatively high probability that two solutions do not dominate each other, which can affect the search performance of a PSO algorithm.

To avoid the above mentioned problem of the dominance-based *pbest* update strategy, we propose a DPU strategy for MOPSO-LS. The DPU strategy is based on TA described in Section 2.2. The update of the *pbest* of each particle is determined based on the aggregated objective function value obtained by TA. However, instead of using the objective function defined in Eq. (4), we use the following modified converting function in MOPSO-LS:

$$g(\mathbf{x}|\boldsymbol{\lambda}^k, \mathbf{z}^*, \mathbf{z}') = \max_{j \in \{1,2\}} \{\lambda_j^k | (f_j(\mathbf{x}) - z_j^*)/(z_j' - z_j^*)|\} \tag{12}$$

where $\boldsymbol{\lambda}^k$ is the weight vector, and $\mathbf{z}^* = (z_1^*, z_2^*)^T$ and $\mathbf{z}' = (z_1', z_2')^T$ are the two reference points mentioned in Section 4.1. We can find that the difference between the modified converting function and the original one is that we use an additional reference point $\mathbf{z}'$, which records the maximum objective function values z_1' and z_2' of solutions in Θ. The normalization operation is embedded in the modified converting function by using the additional reference point $\mathbf{z}'$, since each $f_j(\mathbf{x}) - z_j^*$ is normalized as $(f_j(\mathbf{x}) - z_j^*)/(z_j' - z_j^*)$. This modification can be very beneficial for MOPSO-LS because the ranges of the objective functions f_1 and f_2 in the KQC identification problem are substantially different. The range of f_1 (which measures the classification performance of a QC subset) is $[0, 1]$, while the range of f_2 (which measures the number of selected QCs) is much larger.

Based on the modified TA-based converting function, we obtain the procedure of the DPU strategy, which is shown in Algorithm 2. According to the algorithm, the new position $\mathbf{x}_i^{t+1}$ of each particle is used to update the *pbest* of its neighbors decided by the set $\mathbb{B}_i = \{i_1, i_2, ..., i_T\}$. As shown in line 3 of the algorithm, if we use $\mathbf{x}_i^{t+1}$ to update the *pbest* $\mathbf{p}_j$ of neighbor j, the weight vector $\boldsymbol{\lambda}^j$ of this neighbor is used to compute the aggregated objective function values $g(\mathbf{x}_i^{t+1}|\boldsymbol{\lambda}^j, \mathbf{z}^*, \mathbf{z}')$ and $g(\mathbf{p}_j|\boldsymbol{\lambda}^j, \mathbf{z}^*, \mathbf{z}')$ with Eq.

(12). And we replace the current $\mathbf{p}_j$ with $\mathbf{x}_i^{t+1}$ if $g(\mathbf{x}_i^{t+1}|\boldsymbol{\lambda}^j, \mathbf{z}^*, \mathbf{z}')$ is smaller than $g(\mathbf{p}_j|\boldsymbol{\lambda}^j, \mathbf{z}^*, \mathbf{z}')$. With this strategy, we can update the *pbest*s of all the particles with the new positions.

Algorithm 2: Pseudocode of the DPU strategy.

Input : Reference points $\mathbf{z}^*$ and $\mathbf{z}'$, the *pbest*s $\{\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_N\}$ of particles, new particle positions $\{\mathbf{x}_1^{t+1}, \mathbf{x}_2^{t+1}, ..., \mathbf{x}_N^{t+1}\}$, the uniformly spread vectors $\{\boldsymbol{\lambda}^1, ..., \boldsymbol{\lambda}^N\}$, and neighbor index sets $\mathbb{B}_i = \{i_1, i_2, ..., i_T\}$ $(i = 1, 2, ..., N)$ of particles ;

Output : The updated *pbest*s $\mathbf{p}_i$ $(i = 1, 2, ..., N)$ of particles ;

```
for particle i ← 1, 2, ..., N do
    foreach neighbor j ∈ B_i do
        if g(x_i^{t+1}|λ^j, z*, z') < g(p_j|λ^j, z*, z') then
            p_j ← x_i^{t+1} ;
        end
    end
end
```

4.5. Local search

In MOPSO-LS, the local search is conducted at each iteration to update the non-dominated set Θ after applying the particle swarm based optimization. A solution $\mathbf{x} \in \Theta$ represents a QC/feature subset for the KQC identification problem. For $\mathbf{x}$, two possible operations can be applied for the local search, i.e., the adding and removing operations. The adding operation selects a new QC to $\mathbf{x}$ while the removing operation eliminates an existing QC from $\mathbf{x}$. In this paper, the adding and removing operations are adopted in MOPSO-LS to perform the local search. To perform the adding operation on $\mathbf{x}$, each QC that is not in $\mathbf{x}$ can be selected, which results in $D - |\mathbf{x}|$ ($|\mathbf{x}|$ is the number of QCs selected by $\mathbf{x}$) possible adding operations. Similarly, there are $|\mathbf{x}|$ possible removing operations. To reduce the number of the local search operations at each iteration, only one adding operation and one removing operation are conducted for each $\mathbf{x} \in \Theta$. The adding operation randomly selects a QC into $\mathbf{x}$ and the removing operation randomly eliminates a QC from $\mathbf{x}$.

According to the above analysis, we obtain the mathematical expressions of the adding and removing operations as follows. Let $\mathbf{x} = (x_1, x_2, ..., x_D)^T$ be a solution in Θ, $\mathbb{I}_0 = \{i_1, i_2, ..., i_{(D-|\mathbf{x}|)}\}$ be the index set where each $x_j \leq \tau$ $(j \in \mathbb{I}_0)$, and $\mathbb{I}_1 = \{i_1, i_2, ..., i_{|\mathbf{x}|}\}$ be the index set where each $x_j > \tau$ $(j \in \mathbb{I}_1)$. The adding operation randomly selects an index j from $\mathbb{I}_0$ and generates a new solution based on $\mathbf{x}$ by update its x_j as

$$x_j^{new} = 1 - (1 - \tau) * \frac{x_j}{\tau}. \tag{13}$$

The removing operation randomly selects an index j from $\mathbb{L}_1$ and generates a new solution based on $\mathbf{x}$ by update its x_j as

$$x_j^{new} = \tau * \frac{1 - x_j}{1 - \tau}. \tag{14}$$

5. Experimental design

This section presents the datasets, benchmark methods, and experimental configuration used to verify the proposed MOPSO-LS algorithm.

5.1. Datasets

Four unbalanced production datasets, i.e., PAPER (Wold et al., 2001), ADPN (Gauchi & Chagnon, 2001), LATEX (Gauchi & Chagnon, 2001), and SPIRA (Gauchi & Chagnon, 2001), are used to verify MOPSO-LS. The PAPER dataset was collected from the manufacturing of recycled papers. The QCs that affect the final quality of products include temperatures, flows, concentrations, etc., in the production process. The ADPN dataset was collected from the production process of adiponitrile. The QCs include flows, temperatures, and pressures in the production process. LATEX was collected from the emulsion polymerization batch operations of the production process for latex. The QCs include catalyst levels, reactive concentrations, temperatures, etc. The SPIRA dataset was taken from the fermentation process used to produce an antibiotic. The QCs include temperature levels, oxygen consumption peaks, etc. The original PAPER, ADPN, LATEX, and SPIRA datasets have a continuous response variable measuring the quality of products. Anzanello et al. (2009) converted the continuous response variable for each of these datasets into a discrete class label by the given thresholds in the original studies (Wold et al., 2001; Gauchi & Chagnon, 2001), so that the classification models apply to these datasets. The products in the four converted datasets are classified as the "premium (+1)" and "regular (-1)" quality levels. In the experiments, the four converted datasets are used to verify the proposed MOPSO-LS algorithm. The detailed information of the four datasets is shown in Table 2.

Table 2: The production datasets.

Dataset	Number of products (instances)	Number of premium (+1) products	Number of regular (-1) products	Number of QCs (features)
PAPER	384	33	351	54
ADPN	71	20	51	100
LATEX	262	78	184	117
SPIRA	145	50	95	96

5.2. Benchmark methods

To evaluate the KQC identification performance of MOPSO-LS, six benchmark FS methods, including SFS (Kohavi & John, 1997), SBS (Kohavi & John, 1997), CMDPSOFS (Xue et al., 2013), NSPSOFS (Xue et al., 2013), NSGAII-IPM (Li et al., 2016), and IDMS-IPM (Li et al., 2016), are utilized in the experiments. These methods are based on the same wrapper framework as that used in MOPSO-LS. SFS and SBS are two wrapper methods based on greedy search strategies. SFS searches for the best feature

(QC) subset forwardly by selecting the informative features in the feature subset. SBS searches for the best feature subset backwardly by eliminating uninformative features. CMDPSOFS and NSPSOFS are two wrapper methods that apply two multi-objective PSO algorithms, CMDPSO and NSPSO, to solve a multi-objective FS model. NSGAII-IPM is a wrapper method that utilizes a modified NSGA-II algorithm to solve a multi-objective FS model. This method is proposed for identifying KQCs in production processes. The multi-objective FS model used in CMDPSOFS, NSPSOFS, and NSGAII-IPM is defined as maximizing the accuracy rate and minimizing the number of selected features. IDMS-IPM is a recently proposed wrapper method that adopts the IDMS algorithm for solving an FS model. IDMS-IPM is also proposed for KQC identification. This method adopts the same KQC identification (FS) model as that used in MOPSO-LS. The NSGAII-IPM and DMS-IPM methods utilize the IPM in the second FS phase to select the final KQC set from the non-dominated solutions found by the multi-objective optimization algorithms, which is the same as MOPSO-LS. Similarly, we apply IPM to obtain the final KQC set for CMDPSOFS and NSPSOFS in the experiments for comparisons.

5.3. Experimental configuration

To evaluate the KQC identification performance of MOPSO-LS and the benchmark methods, the 10-fold cross-validation (CV) (Rodriguez et al., 2010) is used to design the experiments. In a 10-fold CV process, the original dataset is divided into 10 folds (denoted by F_1, F_2,..., F_{10}), and 10 pairs of the training set and test set are generated. Specifically, for the ith ($i = 1, 2, ..., 10$) pair, the fold F_i is adopted as the test set $\mathcal{D}_{te}$ and the other 9 folds is combined as the training set $\mathcal{D}_{tr}$. In an experimental run, $\mathcal{D}_{tr}$ is used by the FS methods to select the KQC set, and the unseen $\mathcal{D}_{te}$ is then used to verify the goodness of the obtained KQC set. Thus, a 10-fold CV results in 10 experimental runs. Because the proposed MOPSO-LS and the benchmark methods except for SFS and SBS are based on random search strategies. We repeat the 10-fold CV 3 times (i.e., 30 runs of the experiments) with different random seeds for these FS methods. The results for the 30 runs are used to compare these methods. Note that the FS methods tested in the experiments are based on the wrapper framework that evaluates the importance of a feature subset based on the classification performance measures. In the experiments, we apply the inner 5-fold CV on the training set $\mathcal{D}_{tr}$ to evaluate the classification performance of feature subsets for these FS methods as suggested in Kohavi & John (1997). The learning algorithm used in the experiments is the NB classifier (John & Langley, 1995). This classifier is used as an inner component of the wrapper-based FS methods for evaluating the feature subsets as introduced in Section 3.2. Additionally, in the test phase, the NB classifier is used to evaluate the effectiveness (the classification performance for product quality) of the found KQC sets of the FS methods.

We implement the experiments on a PC with a 3.6 GHz CPU and 16 GB RAM. SFS, SBS, and the NB classifier are implemented based on the Weka (Waikato Environment for Knowledge Analysis) data mining

package (Hall et al., 2009). The FS methods except for SFS and SBS are implemented in Matlab 2021a. It should be noted that the PAPER dataset is highly unbalanced because it includes substantially more regular products than premium products. This can significantly affect the classification performance of the used NB classifier. Therefore, we modify the NB classifier used in PAPER by embedding an up-sampling process as suggested in Li et al. (2020). Specifically, we first balance the training set with the up-sampling strategy by duplicating the positive instances (premium products) for $[\#nIns/\#pIns - 1]$ times ($\#pIns$ and $\#nIns$ denotes the number of positive and negative instances respectively), and then, use the balanced training set to train the NB classifier.

In the experiments, SFS and SBS utilize the default settings in Weka. In CMDPSOFS, NSPSOFS, and NSGAII-IPM, the swarm/population size is set as $N = 100$ as used in Li et al. (2019) and the maximum number of iterations is set as $T_{max} = 100$, which means $10,000$ function evaluations are used. To make a fair comparison, the termination condition in MOPSO-LS and IDMS-IPM is set as $10,000$ function evaluations. Other parameters used in MOPSO-LS, CMDPSOFS, NSPSOFS, and NSGAII-IPM are set as follows. In MOPSO-LS, we set the size of neighbors $T = 10$ (the same as that used in MOEA/D (Zhang & Li, 2007)) and set the mutation probability $p_m = 1/D$. Moreover, we set the inertia weight $w = 0.7298$ and accelerate parameters $c_1 = c_2 = 1.49618$ as suggested in Xue et al. (2013). In CMDPSOFS, the mutation probability is set as $p_m = 1/D$, the inertia weight w is a random number in $[0.1, 0.5]$, c_1 and c_2 are random numbers in $[1.5, 2.0]$ as suggested in Xue et al. (2013). In NSPSOFS, we set $w = 0.7298$ and $c_1 = c_2 = 1.49618$ (Xue et al., 2013). Moreover, in CMDPSO and NSPSOFS, we set the threshold parameter $\tau = 0.6$ and $v_{max} = 0.6$ (Xue et al., 2013). In IDMS-IPM, we set $\alpha^0 = 2$, $\beta_1 = \beta_2 = 0.95$, and $\gamma = 1$ as suggested in Li et al. (2019).

6. KQC identification results and discussion

This section presents the KQC identification results of the proposed MOPSO-LS algorithm and benchmark methods. The metrics used for comparing these methods are accuracy, sensitivity & specificity, GM, the number of selected KQCs, and computational time. The accuracy, sensitivity & specificity, and GM metrics (introduced in Section 3.2) measure the classification performance for product quality of a KQC set, which indicates the effectiveness of these selected KQCs. To calculate the accuracy, sensitivity & specificity, and GM values of an FS method in each experimental run, the classification results on the test set with the learning algorithm built by the selected KQCs are used. The number of selected KQCs indicates the feature reduction performance of these methods. If a method can obtain good classification performance while selecting a few KQCs, this method performs effectively in KQC identification as it effectively eliminates irrelevant and redundant QCs. Because 30 results are obtained for each method from the 30 experimental runs, we compare MOPSO-LS with each benchmark method using the Wilcoxon signed-rank test (Wilcoxon, 1945) on each of the above-mentioned performance metrics.

Tables 3 to 8 show the results of accuracy, sensitivity, specificity, GM, the number of selected KQCs, and computational time obtained by the FS methods on the four production datasets. For each method, the mean and standard deviation (i.e., mean ± standard deviation) of each performance metric over the 30 runs are recorded for each dataset in these tables. The signs "⇑⇑", "↑" , "⇓⇓", and "↓" show the significance test results from the Wilcoxon signed-rank test. The "⇑⇑" or "⇓⇓" indicates the mean value of the performance metric of MOPSO-LS is significantly better or worse than that of the compared method at a significance level of $\alpha = 0.05$ (P-value < 0.05), and the "⇑" or "⇓" indicates the mean value of the performance metric of MOPSO-LS is significantly better or worse than that of the compared method at a significance level of $\alpha = 0.1$ (P-value < 0.1). The detailed analysis based on the results in these tables are given in Sections 6.1 to 6.5 and the discussion is presented in Section 6.6.

6.1. Comparison of the accuracy results

Table 3 shows the accuracy rates of the FS methods. On PAPER, MOPSO-LS obtains a mean accuracy rate of 87.08%, which is slightly lower than the benchmark methods except for SFS. The significance test results show that MOPSO-LS obtains a significantly lower accuracy rate than CMDPSOFS and NSPSOFS, obtains a significantly higher accuracy rate than SFS. On ADPN, MOPSO-LS obtains a mean accuracy rate of 83.75%, which is higher than the mean accuracy rates of all the benchmark FS methods. The significance test results indicate that MOPSO-LS performs significantly better than SFS, SBS, and CMDPSOFS. On LATEX, MOPSO-LS obtains a mean accuracy rate of 79.44%, similar to that of the benchmark methods except for SBS. On SPIRA, MOPSO-LS obtains a mean accuracy rate of 77.52%, significantly higher than that of CMDPSOFS. SFS obtains the best accuracy rate of 82.81%, which is significantly better than MOPSO-LS. Overall, in most cases, the proposed MOPSO-LS can obtain accuracy rates better than or similar to the benchmark methods on the four datasets. This indicates that MOPSO-LS obtains acceptable results in terms of accuracy compared with the benchmark methods.

Table 3: The accuracy rates (%) obtained by the FS methods.

Dataset	MOPSO-LS	SFS	SBS	CMDPSOFS	NSPSOFS	NSGAII-IPM	IDMS-IPM
PAPER	87.08 ± 4.58	82.06 ± 7.40 ⇑⇑	87.99 ± 5.59	89.02 ± 5.24 ⇓⇓	90.15 ± 4.23 ⇓⇓	88.35 ± 4.68	87.52 ± 5.17
ADPN	83.75 ± 8.14	77.32 ± 13.22 ⇑⇑	75.71 ± 14.36 ⇑⇑	76.66 ± 13.30 ⇑⇑	80.88 ± 11.11	81.76 ± 10.14	80.14 ± 13.13
LATEX	79.44 ± 7.86	78.62 ± 6.72	76.75 ± 8.72	79.29 ± 7.94	78.13 ± 6.06	79.27 ± 8.00	79.26 ± 6.37
SPIRA	77.52 ± 10.25	82.81 ± 5.38 ⇓⇓	73.38 ± 11.53	73.68 ± 9.14 ⇑⇑	79.64 ± 7.80	76.65 ± 7.79	76.58 ± 7.58
AVERAGE	81.95	80.20	78.46	79.66	82.20	81.51	80.88

6.2. Comparison of the sensitivity and specificity results

Table 4 shows the sensitivity rates obtained by the FS methods, which reflects the performance of the identified KQCs for classifying the positive instances (i.e., the premium products). On PAPER, MOPSO-LS obtains a mean sensitivity value of 90.00%, which is significantly higher than that of SFS, SBS, CMDPSOFS, NSPSOFS, and NSGAII-IPM. Meanwhile, the mean sensitivity rate obtained by IDMS-IPM is slightly lower

Table 4: The sensitivity rates (%) obtained by the FS methods.

Dataset	MOPSO-LS	SFS	SBS	CMDPSOFS	NSPSOFS	NSGAII-IPM	IDMS-IPM
PAPER	90.00 ± 14.34	58.33 ± 30.28 ↑↑	80.83 ± 20.43 ↑↑	71.94 ± 24.67 ↑↑	74.77 ± 17.72 ↑↑	66.84 ± 25.03 ↑↑	88.61 ± 14.19
ADPN	80.83 ± 27.14	75.00 ± 25.00	65.00 ± 32.02 ↑↑	63.12 ± 37.44 ↑↑	67.50 ± 33.01 ↑↑	75.28 ± 29.38	75.83 ± 26.99
LATEX	72.02 ± 20.24	49.82 ± 17.79 ↑↑	63.93 ± 15.23 ↑	64.88 ± 18.47 ↑↑	57.89 ± 23.86 ↑↑	63.48 ± 21.21 ↑↑	73.27 ± 18.04
SPIRA	67.33 ± 14.13	74.00 ± 15.62 ↓↓	62.00 ± 24.41	61.78 ± 18.45	63.67 ± 20.73	63.56 ± 20.22	68.67 ± 18.39
AVERAGE	77.55	64.29	67.94	65.43	65.96	67.29	76.60

Table 5: The specificity rates (%) obtained by the FS methods.

Dataset	MOPSO-LS	SFS	SBS	CMDPSOFS	NSPSOFS	NSGAII-IPM	IDMS-IPM
PAPER	86.80 ± 4.56	84.36 ± 7.60	88.88 ± 7.17	90.64 ± 4.70 ↓↓	91.59 ± 4.22 ↓↓	90.40 ± 4.71 ↓↓	87.46 ± 5.50
ADPN	84.89 ± 12.26	78.00 ± 20.88 ↑↑	80.00 ± 12.65 ↑↑	82.15 ± 13.40	86.06 ± 12.40	84.20 ± 12.45	81.82 ± 15.66
LATEX	82.65 ± 7.75	90.70 ± 5.63 ↓↓	82.14 ± 9.36	85.34 ± 8.93 ↓	86.60 ± 8.10 ↓↓	85.84 ± 7.99 ↓	81.77 ± 7.38
SPIRA	82.96 ± 14.24	87.56 ± 8.90 ↓	79.22 ± 11.32	79.93 ± 12.17 ↑	88.11 ± 9.73	83.55 ± 11.83	80.71 ± 9.88
AVERAGE	84.33	85.15	82.56	84.52	88.09	86.00	82.94

than that of MOPSO-LS. On ADPN, MOPSO-LS obtains a mean sensitivity rate of 80.83%, which is higher than that of all the benchmark methods, and the difference of the sensitivity results between MOPPSO-LS and SBS/CMDPSOFS/NSPSOFS is significant. On LATEX, MOPSO-LS obtains a mean sensitivity rate of 72.02%, which is significantly higher than the benchmark methods except for IDMS-IPM. IDMS-IPM obtains a slightly better sensitivity rate (i.e., 73.27%) than MOPSO-LS, but no evidence indicates the difference is statistically significant. On SPIRA, MOPSO-LS obtains a mean sensitivity rate of 67.33%, which is lower than that of SFS and IDMS-IPM, and higher than other methods. SFS acquires the best sensitivity rate on SPIRA, and its sensitivity rate is significantly higher than the proposed MOPSO-LS. According to the average results, MOPSO-LS and IDMS-IPM obtain substantially better average sensitivity rates than other methods. This indicates that these two methods can obtain KQCs that have good performance in classifying the minority premium products in the datasets.

Table 5 shows the specificity rates obtained by the FS methods, which reflects the performance of KQCs for classifying the negative instances (i.e., the regular products). On PAPER, MOPSO-LS obtains a mean specificity rate of 86.80%, which is significantly lower than that of CMDPSOFS, NSPSOFS, and NSGAII-IPM, close to that of SBS and IDMS-IPM, and higher than that of SFS. On ADPN, MOPSO-LS obtains a mean specificity rate of 84.89%, which is significantly higher than that of SFS and SBS. The mean specificity rate of MOPSO-LS is only slightly lower than NSPSOFS, but no evidence shows that the difference between MOPSO-LS and NSPSOFS is significant. On LATEX, MOPSO-LS obtains a specificity rate of 82.65%, which is similar to SBS and IDMS-IPM. Meanwhile, the specificity rate of MOPSO-LS is significantly lower than that of SFS, CMDPSOFS, NSPSOFS, and NSGAII-IPM. On SPIRA, MOPSO-LS obtains a specificity rate of 82.96%, which is higher than or similar to that of the benchmark methods except for SFS and NSPSOFS.

Overall, the above results indicate that MOPSO-LS obtains significantly better sensitivity results in most cases. Meanwhile, MOPSO-LS does not obtain significantly better specificity rates than benchmark

Table 6: The GM values (%) obtained by the FS methods.

Dataset	MOPSO-LS	SFS	SBS	CMDPSOFS	NSPSOFS	NSGAII-IPM	IDMS-IPM
PAPER	88.09 ± 8.06	65.36 ± 25.44 ↑↑	83.68 ± 9.58 ↑↑	78.47 ± 19.51 ↑↑	81.70 ± 11.12 ↑↑	74.44 ± 22.22 ↑↑	87.70 ± 7.79
ADPN	79.64 ± 18.46	73.78 ± 12.54 ↑↑	67.51 ± 27.25 ↑↑	62.58 ± 33.31 ↑↑	69.80 ± 27.68 ↑	75.26 ± 23.28	75.77 ± 20.43
LATEX	75.84 ± 13.52	66.12 ± 12.12 ↑↑	71.92 ± 10.54 ↑	73.33 ± 11.62	68.55 ± 14.15 ↑↑	72.27 ± 14.32	76.58 ± 9.86
SPIRA	74.02 ± 9.98	79.72 ± 7.00 ↓↓	66.09 ± 23.91	69.12 ± 10.72 ↑↑	73.33 ± 12.71	70.42 ± 15.52	73.35 ± 10.42
AVERAGE	79.40	71.24	72.30	70.88	73.35	73.09	78.35

methods in most cases. In some cases, MOPSO-LS obtains slightly worse specificity rates than benchmark methods. This shows that MOPSO-LS prefers to improve the classification performance for the minority premium quality products compared with the benchmark methods (except for IDMS-IPM). Moreover, we find that, compared with MOPSO-LS, the sensitivity rates of benchmark methods (except for IDMS-IPM) are generally substantially lower than the specificity rates. This indicates these methods do not perform well in selecting the real KQCs that effectively predict the quality of both the premium and regular products. Different from other benchmark methods, IDMS-IPM shows similar performance on the sensitivity and specificity metrics compared with MOPSO-LS. This indicates the multi-objective KQC identification model used in MOPSO-LS and IDMS-IPM is more effective than that of other methods. In the next section, we will use an integrated measure of sensitivity and specificity, i.e., GM, to further illustrate the effectiveness of MOPSO-LS.

6.3. Comparison of the GM results

Table 6 shows the GM values obtained by the FS methods. GM is a widely used measure for evaluating classification performance on unbalanced data. A high GM value indicates the KQCs identified by an FS method have good classification performance for both the premium and regular products. Thus, GM can reflect the KQC identification performance.

On PAPER, MOPSO-LS obtains a GM value of 88.09%, which is significantly higher than that of the benchmark methods except for IDMS-IPM. On ADPN, MOPSO-LS obtains a mean GM value of 79.64%, which is significantly higher than that of SFS, SBS, CMDPSOFS, and NSPSOFS. Meanwhile, MOPSO-LS obtains a higher mean GM value than NSGAII-IPM and IDMS-IPM, but the significance test results are not significant. On LATEX, MOPSO-LS obtains a higher mean GM value (75.84%) than all the benchmark methods, and the difference between MOPSO-LS and SFS/SBS/NSPSOFS is significant. On SPIRA, SFS obtains the best mean GM value. Meanwhile, the mean GM value of MOPSO-LS is higher than the benchmark methods except for SFS. According to the average results over the four datasets, MOPSO-LS obtains a substantially higher average GM value than the benchmark methods except for IDMS-IPM. The above results indicate that MOPSO-LS generally obtains significantly better GM values than the benchmark methods except for IDMS-IPM, which shows its effectiveness for KQC identification on the unbalanced production data.

Table 7: The number of selected KQCs of each FS method.

Dataset	MOPSO-LS	SFS	SBS	CMDPSOFS	NSPSOFS	NSGAII-IPM	IDMS-IPM
PAPER	2.77 ± 0.50	4.00 ± 1.79 ↑↑	18.20 ± 9.82 ↑↑	4.67 ± 1.40 ↑↑	5.37 ± 1.58 ↑↑	4.80 ± 0.87 ↑↑	3.20 ± 0.87 ↑↑
ADPN	2.43 ± 0.56	5.30 ± 1.27 ↑↑	25.80 ± 9.87 ↑↑	6.07 ± 1.75 ↑↑	3.47 ± 1.28 ↑↑	2.80 ± 0.60 ↑↑	15.93 ± 6.45 ↑↑
LATEX	3.67 ± 0.91	7.70 ± 2.90 ↑↑	93.80 ± 15.85 ↑↑	10.37 ± 2.44 ↑↑	5.00 ± 1.81 ↑↑	5.87 ± 2.17 ↑↑	30.00 ± 4.85 ↑↑
SPIRA	3.57 ± 0.96	3.50 ± 1.02	51.70 ± 16.51 ↑↑	6.60 ± 1.85 ↑↑	3.57 ± 1.20	4.07 ± 1.26 ↑	17.83 ± 4.71 ↑↑
AVERAGE	3.11	5.12	47.38	6.92	4.35	4.38	16.74

6.4. Comparison of the number of selected KQCs

Table 7 shows the number of selected KQCs obtained by the FS methods. MOPSO-LS selects significantly fewer KQCs than all the benchmark methods on PAPER, ADPN, and LATEX, and selects significantly fewer KQCs than the benchmark methods except for SFS and NSPSOFS on SPIRA. No evidence indicates that MOPSO-LS selects significantly more KQCs than the benchmark methods. These results show that MOPSO-LS has much better feature reduction performance than the benchmark methods. Moreover, without considering MOPSO-LS, SFS, CMDPSOFS, NSPSOFS, and NSGAII-IPM perform the best on feature reduction and the average number of selected KQCs of these methods is around 5. SBS performs the worst on feature reduction as the average number of selected KQCs is 47.38, which is substantially larger than that of other methods. This is because the greedy search nature of SBS makes it easily trapped into local optima (i.e., QC subset with a large size). The feature reduction performance of IDMS-IPM is between SBS and other methods as it selects 16.74 KQCs on average.

To further evaluate the feature reduction performance of MOPSO-LS, the results of the number of selected QCs during the iterations are drawn in Fig. 3 for MOPSO-LS and the benchmark multi-objective FS methods, i.e., CMDPSOFS, NSPSOFS, NSGAII-IPM, and IDMS-IPM. Because all these FS methods are multi-objective approaches, the best "solution" at each iteration is the non-dominated set instead of a single solution. To facilitate the comparison, we use the median of the number of selected QCs for the non-dominated solutions found at each iteration as the metric to compare the methods. Moreover, as 30 experimental runs are conducted, the shown curves in Fig. 3 are drawn with the average value of the 30 medians from the 30 runs. According to Fig. 3, MOPSO-LS shows better feature reduction performance than the benchmark methods except for NSPSOFS during the whole evolutionary process since the curves of MOPSO-LS are lower than these methods on the four datasets. On PAPER, compared with NSPSOFS, MOPSO-LS obtains a lower curve, which denotes that MOPSO-LS shows better feature reduction performance. On ADPN, LATEX, and SPIRA, although MOPSO-LS reduces the number of selected QCs slower than NSPSOFS at the early stage of the evolutionary process, MOPSO-LS can obtain lower curves than NSPSOFS when the number of function evaluations is larger than around 2,000. It is also found that MOPSO-LS, CMDPSO, and NSPSOFS start with a smaller value of the number of selected QCs than NSGAII-IPM and IDMS-IPM. This is because MOPSO-LS, CMDPSO, and NSPSOFS adopt a threshold value $\tau = 0.6$ to determine the selection of QCs for each encoded particle in the swarm, which

means less than 50% QCs are expected to be selected by the initialized particles of the three methods. Overall, the results in Table 7 and Fig. 3 indicate that MOPSO-LS has good feature reduction performance.

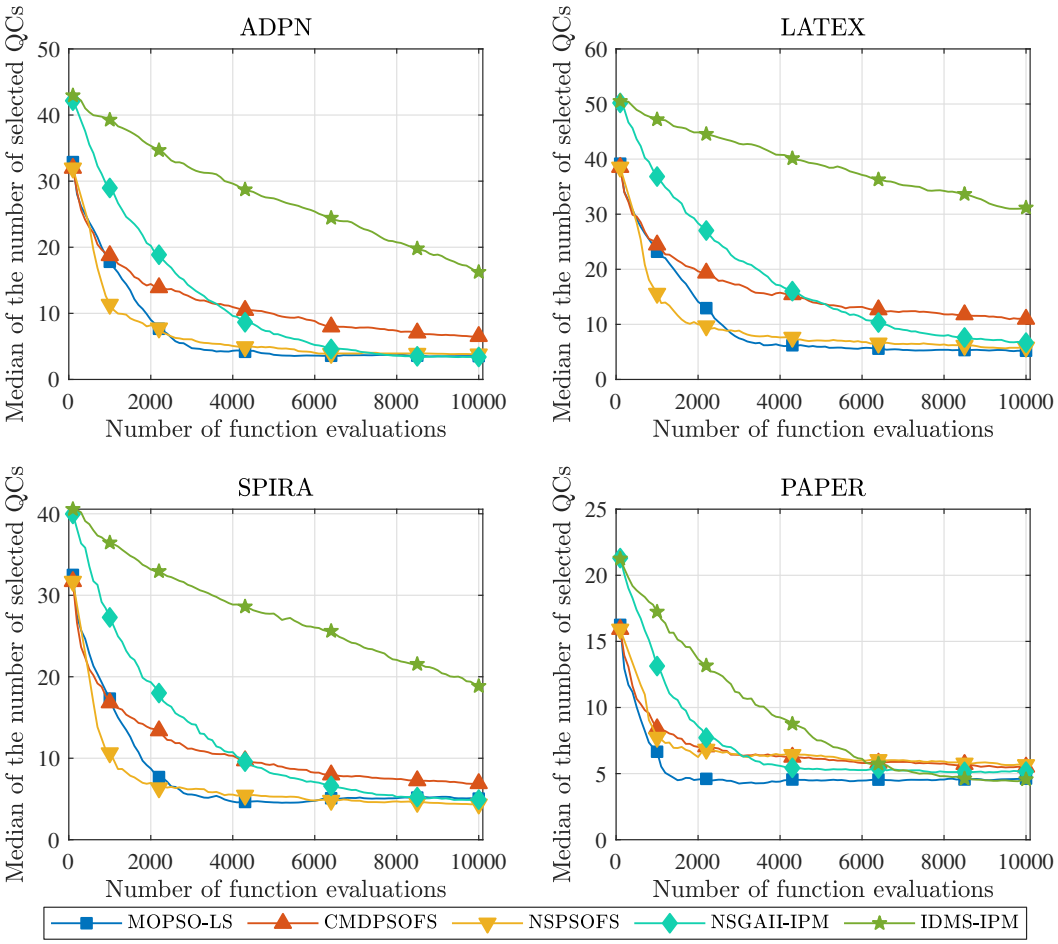

Figure 3: The number of selected QCs during the iterations for the multi-objective FS methods.

6.5. Comparison of the computational time

Table 8 shows the mean and standard deviation of the computational time over 30 runs of each method on each dataset. Overall, SFS is the most time efficient as it consumes much less time than other methods on the four datasets. SBS costs substantially more time than other methods on the datasets except for PAPER. The computational time of MOPSO-LS is close to the benchmark methods except for SFS and SBS on the four datasets. The reason for the similar computational time between MOPSO-LS and these benchmark methods is that they terminate the algorithm with the same number of function evaluations. These benchmark methods require similar computational time to MOPSO-LS for function evaluations, which is the most time consuming part of these wrapper-based FS methods. Overall, the above results indicate that

the improvement of the KQC identification performance of MOPSO-LS does not require more computational resources.

Table 8: Comparison of computational time (in minutes) of the FS methods

Dataset	MOPSO-LS	SFS	SBS	CMDPSOFS	NSPSOFS	NSGAII-IPM	IDMS-IPM
PAPER	7.04 ± 0.52	0.20 ± 0.11 ↓↓	7.36 ± 1.10 ↑↑	7.51 ± 0.36 ↑↑	7.03 ± 0.68	6.84 ± 0.24 ↓	4.85 ± 0.52 ↓↓
ADPN	1.80 ± 0.13	0.17 ± 0.04 ↓↓	5.47 ± 0.69 ↑↑	1.52 ± 0.06 ↓↓	1.38 ± 0.08 ↓↓	1.44 ± 0.04 ↓↓	1.45 ± 0.12 ↓↓
LATEX	7.94 ± 0.68	1.04 ± 0.53 ↓↓	30.65 ± 13.86 ↑↑	9.49 ± 0.67 ↑↑	7.39 ± 1.19 ↓↓	8.60 ± 0.62 ↑↑	9.53 ± 0.75 ↑↑
SPIRA	3.49 ± 0.26	0.18 ± 0.06 ↓↓	13.59 ± 2.04 ↑↑	3.35 ± 0.20 ↓↓	2.87 ± 0.26 ↓↓	3.14 ± 0.15 ↓↓	3.30 ± 0.32 ↓↓
AVERAGE	5.07	0.40	14.27	5.47	4.67	5.00	4.78

6.6. Discussion

The results in Sections 6.1 to 6.3 indicate the identified KQCs of the proposed MOPSO-LS obtain good classification performance on the four unbalanced production datasets. Compared with the benchmark methods, MOPSO-LS obtains similar or better accuracy rates and obtains significantly better GM results in most cases. The comparisons in terms of sensitivity and specificity reveal that MOPSO-LS obtains good classification performance for both the minority premium and majority regular products. These results indicate that MOPSO-LS properly selects the KQCs that can effectively predict the quality of products. Further comparisons of the number of selected KQCs show that MOPSO-LS performs effectively on feature reduction as the number of selected KQCs of MOPSO-LS is generally very small on each dataset. Therefore, MOPSO-LS is effective for identifying KQCs on these unbalanced production datasets. The effectiveness of MOPSO-LS can be explained by the following two reasons:

- The KQC identification model adopted by MOPSO-LS is effective on the unbalanced production data. MOPSO-LS and IDMS-IPM adopt the same model that aims at maximizing the GM value and minimizing the number of selected QCs. A high GM value requires good classification performance on both premium and regular products. It is a better classification performance than the accuracy used in other benchmark methods for unbalanced data. Therefore, by optimizing this model, the KQCs with a good classification capability for product quality can be selected by MOPSO-LS and IDMS-IPM.

- The proposed MOPSO-LS has good search performance for solving the KQC identification model. The results in Sections 6.1 to 6.3 have shown that MOPSO-LS selects substantially fewer KQCs while obtaining better classification results than IDMS-IPM even though these two methods adopt the same KQC identification model. This indicates that MOPSO-LS has better search performance than IDMS-IPM. However, since CMDPSOFS, NSPSOFS, and NSGAII-IPM utilize a model different from MOPSO-LS, the search performance is not comparable between the three benchmark methods and MOPSO-LS. Therefore, in Section 7, we further compare the search performance of MOPSO-LS and benchmark optimization algorithms with additional experiments.

The proposed MOPSO-LS algorithm is a data-driven approach that utilizes the data collected from the production process for selecting KQCs related to the quality of final products. It can be used to identify the potential factors that cause the failure of production processes. The identified KQCs can be used to build an effective quality prediction model. Such a quality prediction model can be used to build a virtual metrology (VM) (Chien et al., 2022) or soft sensor (Feng et al., 2021) method to improve the efficiency in measuring product quality, which can substantially improve the efficiency and quality of production processes. Furthermore, since the proposed MOPSO-LS algorithm helps in the control of production processes by using the production data, it follows the basic principles of total quality management (TQM) such as process-centered approach and fact-based decision making. Therefore, MOPSO-LS can be a useful quality engineering tool to help manufacturing enterprises implement the TQM.

There are also some limitations in the proposed MOPSO-LS algorithm. First, we have applied MOPSO-LS to four datasets from different production processes to verify its performance. For each production process (e.g., the paper recycling process), only one dataset is applied to decide the KQCs. From a practical point of view, it is required to use different datasets from the same production process to further confirm if the identified KQCs of MOPSO-LS are the real key factors affecting product quality. Due to a lack of available datasets, we have not done this type of experiment to further validate the identified KQCs. It is valuable to verify the robustness and effectiveness of existing data-driving KQC identification approaches on different datasets from the same production process in the future. This is also beneficial for confirming the correctness of the KQC identification results for the production process. Second, the data-driven feature of MOPSO-LS makes it a general approach applicable to different types of production processes. However, one limitation of MOPSO-LS is that it does not apply any domain knowledge in KQC identification. Embedding domain knowledge in MOPSO-LS is beneficial for further improving its performance for a particular production process. This would be another valuable topic worth to be studied in the future.

7. Further analysis of the search performance

In Section 6, we have verified the KQC identification performance of MOPSO-LS. As we discussed, the effectiveness of the applied KQC identification model for the unbalanced production data is one reason for the effectiveness of MOPSO-LS. Another possible reason is that MOPSO-LS has good search performance. In this section, we will analyze the search performance of MOPSO-LS in detail.

7.1. Benchmark algorithms

We compare the search performance of MOPSO-LS with the search algorithms used in the benchmark multi-objective FS methods introduced in Section 5, which include CMDPSOFS, NSPSOFS, NSGAII-IPM, and IDMS-IPM. We denote the search algorithms used in these FS methods by CMDPSO, NSPSO, MNSGA-II (a modified NSGA-II), and IDMS. Compared with MOPSO-LS, the benchmark algorithms CMDPSO,

NSPSO, and MNSGAII applied a different KQC identification model, which maximizes the accuracy (instead of GM) and minimizes the number of selected features, in Section 6. Therefore, we further conduct experiments for these algorithms on the same KQC identification model as MOPSO-LS to facilitate the comparison. The experimental settings and the datasets are the same as that described in Section 5, i.e., 3 repetitions of 10-fold CV are conducted on each dataset for each method and 30 sets of optimization results are obtained. Note that, since we aim to evaluate the search performance of these multi-objective optimization algorithms, the final solution selected by IPM is not used for comparison. Instead, the overall performance of all the non-dominated solutions found by these multi-objective optimization algorithms is considered to evaluate the performance of these optimization algorithms. The performance metrics that evaluate the search performance are introduced below.

7.2. Performance metrics

We adopt three widely used performance metrics (also known as quality indicators), including Hypervolume (HV) (Yuan et al., 2016; Li et al., 2022), inverted generational distance (IGD) (Deb & Jain, 2014; Li et al., 2022), and set coverage (SC) (Zitzler & Thiele, 1999), to evaluate the search performance of a multi-objective optimization algorithm. A brief description of the three metrics is given below:

- *The HV metric*: Let $\mathbb{A}$ be the set of non-dominated points (here non-dominated points refer to the non-dominated solutions in the objective space) returned by an optimization algorithm, and $\mathbf{r} = (r_1, ..., r_m)^T$ (m is the number of objectives) be a reference point in the objective space. The HV value of $\mathbb{A}$ given the reference point $\mathbf{r}$ is obtained as

$$HV(\mathbb{A}, \mathbf{r}) = volume\Big(\bigcup_{\mathbf{a} \in \mathbb{A}} \prod_{j=1}^{m} |a_j - r_j| \Big) \tag{15}$$

  where $\mathbf{a} = (a_1, ..., a_m)$ is a non-dominated point in $\mathbb{A}$ and each a_j denotes the value of the jth objective function. HV measures the volume of the region dominated by a non-dominated set in the objective space. It is a Pareto-compliant performance metric that has good theoretical qualities for fairly comparing the multi-objective optimization algorithms. A larger $HV(\mathbb{A}, \mathbf{r})$ value denotes a better performance of the optimization algorithm.

- *The IGD metric*: Let $\mathbb{A}$ be the set of non-dominated points returned by an optimization algorithm and $\mathbb{P}$ be the PF (the set of Pareto solutions in the objective space). IGD measures the similarity between $\mathbb{A}$ and $\mathbb{P}$ as

$$IGD(\mathbb{A}, \mathbb{P}) = \frac{1}{|\mathbb{P}|} \sum_{\mathbf{p} \in \mathbb{P}} \min_{\mathbf{a} \in \mathbb{A}} d(\mathbf{a}, \mathbf{p}) \tag{16}$$

  where $d(\mathbf{a}, \mathbf{p})$ denotes the Euclidean distance between $\mathbf{a}$ and $\mathbf{p}$, and $|\mathbb{P}|$ denotes the number of points in $\mathbb{P}$. IGD can measure both the convergence and diversity performance of the non-dominated solutions

found by an algorithm. A smaller IGD value denotes a higher similarity level between $\mathbb{A}$ and $\mathbb{P}$, which denotes a better performance of the optimization algorithm.

- *The SC metric*: Let $\mathbb{A}$ and $\mathbb{P}$ be two sets of non-dominated points. The value of $SC(\mathbb{A}, \mathbb{P})$ is defined as the percentage of points in $\mathbb{P}$ that are dominated by or equal to a point in $\mathbb{A}$, i.e.,

$$SC(\mathbb{A}, \mathbb{P}) = \frac{\{\mathbf{p} \in \mathbb{P} | \exists \mathbf{a} \in \mathbb{A} : \mathbf{a} \text{ dominates or equals } \mathbf{p}\}}{|\mathbb{P}|}. \tag{17}$$

  We let $\mathbb{A}$ be the set of non-dominated points returned by an optimization algorithm and $\mathbb{P}$ be the PF. $SC(\mathbb{A}, \mathbb{P})$ can evaluate the goodness of $\mathbb{A}$ by using $\mathbb{P}$ as the benchmark. By calculating $SC(\mathbb{A}, \mathbb{P})$ for each of the optimization algorithms, the performance of these algorithms can be compared with the SC metric. A larger SC value (i.e., $SC(\mathbb{A}, \mathbb{P})$) denotes a better performance of the optimization algorithm.

Since the KQC identification model in Eq. (8) is applied, the objective functions are $f_1 = 1 - GM(\mathbb{X})$ and $f_2 = |\mathbb{X}|$. Before calculating the HV, IGD, and SC values for each algorithm, we first normalize the two objective functions with the min-max normalization method to [0,1] since the ranges of these two objective functions differ a lot. For each run, the maximum/minimum value used in the normalization is obtained from the solutions returned by all the algorithms at the 3 repetitions of the experiments. The normalized objective function values are used to obtain the HV, IGD, and SC results. With the normalization, more reasonable results can be obtained to compare these optimization algorithms by balancing the weights of the two objectives. The HV metric requires a given reference point, which is set as $\mathbf{r} = (1.1, 1.1)$ as suggested by Yuan et al. (2016). The IGD and SC metrics require a known PF $\mathbb{P}$. However, the true PFs of the KQC identification problems in the experiments are not known. Therefore, for each fold, we first combine all the solutions of these algorithms found from the 3 repetitions of experiments, and then, obtain an approximating PF $\mathbb{P}'$ by selecting the non-dominated solutions from the combined solutions. Based on $\mathbb{P}'$, we can obtain $IGD(\mathbb{A}, \mathbb{P}')$ and $SC(\mathbb{A}, \mathbb{P}')$ to evaluate the search performance of each algorithm. Since 30 sets of the results are obtained for each algorithm on each dataset from the 30 runs, we use the Wilcoxon signed-rank test (Wilcoxon, 1945) to verify if MOPSO-LS performs significantly better than the benchmark algorithms on each of the performance metrics.

7.3. Comparison results and discussion

Table 9 shows the results of the performance metrics of MOPSO-LS and the benchmark algorithms. "$\uparrow\uparrow$" ("$\uparrow$") or "$\downarrow\downarrow$" ("$\downarrow$") shows the Wilcoxon signed-rank test result which indicates MOPSO-LS performs significantly better or worse than the compared algorithm at a significance level of $\alpha = 0.05$ ($\alpha = 0.1$) with respect to the mean value of a performance metric.

Table 9: Comparison of search performance between MOPSO-LS and benchmark algorithms.

Metric	Dataset	MOPSO-LS	CMDPSO	NSPSO	MNSGA-II	IDMS
HV	PAPER	1.117 ± 0.025	1.094 ± 0.029 ⇈	1.096 ± 0.038 ⇈	$\mathbf{1.124 \pm 0.028}$ ⇊	1.096 ± 0.040 ⇈
	ADPN	$\mathbf{1.142 \pm 0.031}$	1.006 ± 0.068 ⇈	1.106 ± 0.045 ⇈	1.121 ± 0.038 ⇈	0.686 ± 0.222 ⇈
	LATEX	$\mathbf{1.135 \pm 0.039}$	0.961 ± 0.046 ⇈	1.062 ± 0.071 ⇈	1.089 ± 0.038 ⇈	0.468 ± 0.113 ⇈
	SPIRA	$\mathbf{1.087 \pm 0.055}$	0.940 ± 0.066 ⇈	1.052 ± 0.052 ⇈	1.062 ± 0.061 ↑	0.554 ± 0.153 ⇈
IGD	PAPER	0.058 ± 0.038	0.062 ± 0.026	0.063 ± 0.037	$\mathbf{0.034 \pm 0.020}$ ⇊	0.073 ± 0.042 ↑
	ADPN	$\mathbf{0.042 \pm 0.022}$	0.141 ± 0.040 ⇈	0.074 ± 0.027 ⇈	0.075 ± 0.033 ⇈	0.378 ± 0.176 ⇈
	LATEX	$\mathbf{0.038 \pm 0.022}$	0.125 ± 0.028 ⇈	0.079 ± 0.038 ⇈	0.056 ± 0.017 ⇈	0.501 ± 0.096 ⇈
	SPIRA	$\mathbf{0.053 \pm 0.020}$	0.127 ± 0.031 ⇈	0.075 ± 0.024 ⇈	0.065 ± 0.030	0.385 ± 0.125 ⇈
SC	PAPER	0.304 ± 0.143	0.144 ± 0.137 ⇈	0.162 ± 0.124 ⇈	$\mathbf{0.373 \pm 0.210}$	0.193 ± 0.179 ⇈
	ADPN	$\mathbf{0.363 \pm 0.201}$	0.013 ± 0.039 ⇈	0.060 ± 0.089 ⇈	0.249 ± 0.253 ⇈	0.000 ± 0.000 ⇈
	LATEX	$\mathbf{0.334 \pm 0.149}$	0.003 ± 0.014 ⇈	0.059 ± 0.125 ⇈	0.107 ± 0.146 ⇈	0.000 ± 0.000 ⇈
	SPIRA	$\mathbf{0.298 \pm 0.186}$	0.000 ± 0.000 ⇈	0.042 ± 0.082 ⇈	0.149 ± 0.159 ⇈	0.003 ± 0.018 ⇈

According to Table 9, MOPSO-LS obtains significantly better results on the three performance metrics in most cases. In terms of HV, MOPSO-LS obtains significantly higher HV values than CMDPSO, NSPSO, and IDMS on all four datasets at a significance level of $\alpha = 0.05$. MOPSO-LS also obtains significantly higher HV values than MNSGA-II on ADPN and LATEX at a significance level of 0.05 and a significantly higher HV value than MNSGA-II on SPIRA at a significance level of $\alpha = 0.1$. The HV value obtained by MOPSO-LS is significantly lower than MNSGA-II on PAPER. In terms of IGD, MOPSO-LS obtains significantly lower IGD values than CMDPSO, NSPSO, and IDMS on three of the four datasets, i.e., ADPN, LATEX, and SPIRA, at a significance level of $\alpha = 0.05$. Compared with MNSGA-II, MOPSO-LS obtains significantly lower IGD values on two datasets, i.e., ADPN and LATEX, and a significantly higher IGD value on PAPER. In terms of SC, MOPSO-LS obtains significantly higher SC values than CMDPSO, NSPSO, and IDMS on all the four datasets at a significance level of $\alpha = 0.05$. MOPSO-LS also obtains significantly higher SC values than MNSGA-II on three of the four datasets, i.e., ADPN, LATEX, and SPIRA.

Overall, the above results indicate that MOPSO-LS obtains significantly better search performance than CMDPSO, NSPSO, and IDMS because MOPSO-LS obtains better results on at least one performance metric on the four datasets. Comparing MOPSO-LS with MNSGA-II, MOPSO-LS obtains better search performance on three datasets (the datasets except for PAPER) and obtains worse search performance on one dataset, i.e., PAPER. The possible reason for this result is related to the nature of GA and PSO. We can find that the PAPER dataset has the smallest number of original QCs among the four datasets. So, given the same number of iterations, the optimization will be more sufficiently conducted on PAPER than on other datasets. Being a PSO algorithm, MOPSO-LS has the advantage of quickly converging to satisfied solutions, while it does not work as effectively as MNSGA-II, a GA, to escape from local optima during the late phase of the optimization. Therefore, for the relatively simple KQC identification problem on PAPER, MNSGA-II obtains better search performance than MOPSO-LS. However, on high dimensional data with a large number of QCs, which requires substantial computational resources, MOPSO-LS can obtain more desirable

optimization results than MNSGA-II since MOPSO-LS has a faster convergence speed than MNSGA-II.

Two possible factors contribute to the good search ability of MOPSO-LS. First, MOPSO-LS adopts the DPU strategy to update *pbest*. The good solutions on different regions of the non-dominated font can be maintained by the *pbest*s of the particles because different particles adopt different weights for the two optimization objectives to construct the aggregated objective function $g(\mathbf{x}_i^{t+1}|\boldsymbol{\lambda}^j, \mathbf{z}^*, \mathbf{z}')$, the criterion used to update *gbest*. Second, the local search step utilized by MOPSO-LS is effective for improving the search performance. The local search step is adopted to just update the non-dominated set after the particle swarm based optimization step at each iteration. Thus, the convergence speed of MOPSO-LS is improved. Moreover, the updated non-dominated set by the local search step provides *gbest* candidates for the particle swarm based optimization step at following iterations, which further improves the search performance of MOPSO-LS.

8. Conclusions

Identifying the KQCs in production processes that are strongly related to the quality of final products is essential for quality control. In this paper, we have proposed an FS method to identify KQCs with good predictive performance for product quality. The proposed method is based on a KQC identification model for unbalanced production data, i.e., a multi-objective optimization model of maximizing the GM measure and minimizing the number of selected QCs. To solve this model, we propose a multi-objective optimization algorithm, called MOPSO-LS, that combines the search mechanisms of PSO and the local search to obtain a set of non-dominated solutions (candidate KQC sets), and then use the IPM to select the final KQC set. In MOPSO-LS, the decomposition approach, i.e., a modified TA, is adopted to update the *pbest*s, which can maintain diversified and good performance solutions. The local search step purifies the non-dominated solutions at each iteration with two basic operations, i.e., adding and removing a feature. We have verified the KQC identification performance of the proposed MOPSO-LS using four production datasets. The results indicate that MOPSO-LS can obtain a KQC set with fewer QCs and better predictive performance for product quality compared with six benchmark FS methods. Specifically, the predictive performance of MOPSO-LS on the minority premium products is significantly improved compared with most of the benchmark methods. Meanwhile, MOPSO-LS requires similar computational time compared with the benchmark multi-objective FS methods. Further analysis indicates that MOPSO-LS has good search performance in optimizing the multi-objective KQC identification model. These results show that MOPSO-LS is effective and efficient for KQC identification. In real quality control and management applications, the proposed MOPSO-LS algorithm can help quality engineers find out the key factors that cause the failure of production processes. The identified KQCs can be used to build an effective quality prediction model for advanced quality control, which helps manufacturing enterprises to improve the competitiveness of their

products.

The proposed KQC identification method can be further extended in the following aspects. First, four datasets from different production processes are utilized to evaluate the performance of the proposed method in the experiments. It would be interesting to further verify the robustness of the method and the correctness of the KQC identification results with more datasets from the same production process. This is also beneficial for providing more reliable and useful information with quality engineers for controlling and improving a particular production process. Second, it is worth embedding domain knowledge in the proposed data-driven KQC identification method to further improve its performance. Finally, the main objective of the proposed method is to identify the key factors (KQCs) in the production process that significantly affect product quality. It is worth building a VM method with effective prediction models using these identified KQCs for advanced quality control. Proposing a proactive control strategy in the VM method to dynamically adjust the KQCs for producing conforming products is required.

Acknowledgments

This work was supported by the Tianjin transportation science and technology development project plan of Tianjin Municipal Transportation Commission [grant number 2022-39]; the National Natural Science Foundation of China (NSFC) [grant number 72101182]; and the Humanities and Social Sciences Youth Fund of Ministry of Education of China [grant numbers 19YJC630071, 19YJC630221].

References

Aljarah, I., Habib, M., Faris, H., Al-Madi, N., Heidari, A. A., Mafarja, M., Elaziz, M. A., & Mirjalili, S. (2020). A dynamic locality multi-objective salp swarm algorithm for feature selection. *Computers & Industrial Engineering*, *147*, 106628.

Amaldi, E., & Kann, V. (1998). On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems. *Theoretical Computer Science*, *209*(1), 237 – 260.

Amoozegar, M., & Minaei-Bidgoli, B. (2018). Optimizing multi-objective PSO based feature selection method using a feature elitism mechanism. *Expert Systems with Applications*, *113*, 499 – 514.

Anzanello, M. J., Albin, S. L., & Chaovalitwongse, W. A. (2009). Selecting the best variables for classifying production batches into two quality levels. *Chemometrics and Intelligent Laboratory Systems*, *97*(2), 111 – 117.

Anzanello, M. J., Albin, S. L., & Chaovalitwongse, W. A. (2012). Multicriteria variable selection for classification of production batches. *European Journal of Operational Research*, *218*(1), 97 – 105.

Chien, C.-F., Hung, W.-T., Pan, C.-W., & Van Nguyen, T. H. (2022). Decision-based virtual metrology for advanced process control to empower smart production and an empirical study for semiconductor manufacturing. *Computers & Industrial Engineering*, *169*, 108245.

Chien, C.-F., Liu, C.-W., & Chuang, S.-C. (2017). Analysing semiconductor manufacturing big data for root cause detection of excursion for yield enhancement. *International Journal of Production Research*, *55*(17), 5095–5107.

Deb, K., & Jain, H. (2014). An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints. *IEEE Transactions on Evolutionary Computation*, *18*(4), 577–601.

Feng, L., Zhao, C., & Sun, Y. (2021). Dual attention-based encoder–decoder: A customized sequence-to-sequence learning for soft sensor development. *IEEE Transactions on Neural Networks and Learning Systems*, *32*(8), 3306–3317.

Gauchi, J.-P., & Chagnon, P. (2001). Comparison of selection methods of explanatory variables in pls regression with application to manufacturing process data. *Chemometrics and Intelligent Laboratory Systems*, *58*(2), 171 – 193.

Guan, B., Zhao, Y., Yin, Y., & Li, Y. (2021). A differential evolution based feature combination selection algorithm for high-dimensional data. *Information Sciences*, *547*, 870–886.

Guo, W., & Banerjee, A. G. (2017). Identification of key features using topological data analysis for accurate prediction of manufacturing system outputs. *Journal of Manufacturing Systems*, *43*, 225–234.

Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research*, *3*, 1157–1182.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, *11*(1), 10–18.

Han, F., Chen, W.-T., Ling, Q.-H., & Han, H. (2021). Multi-objective particle swarm optimization with adaptive strategies for feature selection. *Swarm and Evolutionary Computation*, *62*, 100847.

Ishibuchi, H., & Murata, T. (1998). A multi-objective genetic local search algorithm and its application to flowshop scheduling. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *28*(3), 392–403.

Jeong, B., & Cho, H. (2006). Feature selection techniques and comparative studies for large-scale manufacturing processes. *The International Journal of Advanced Manufacturing Technology*, *28*(9), 1006–1011.

Jin, N., & Zhou, S. (2006). Data-driven variation source identification for manufacturing process using the eigenspace comparison method. *Naval Research Logistics (NRL)*, *53*(5), 383–396.

John, G. H., & Langley, P. (1995). Estimating continuous distributions in bayesian classifiers. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence* UAI'95 (pp. 338–345). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Kang, Z., Catal, C., & Tekinerdogan, B. (2020). Machine learning applications in production lines: A systematic literature review. *Computers & Industrial Engineering*, *149*, 106773.

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks* (pp. 1942–1948 vol.4). volume 4.

Kiziloz, H. E., & Deniz, A. (2021). An evolutionary parallel multiobjective feature selection framework. *Computers & Industrial Engineering*, *159*, 107481.

Kohavi, R., & John, G. H. (1997). Wrappers for feature subset selection. *Artificial Intelligence*, *97*(1), 273 – 324.

Kozodoi, N., Lessmann, S., Papakonstantinou, K., Gatsoulis, Y., & Baesens, B. (2019). A multi-objective approach for profit-driven feature selection in credit scoring. *Decision Support Systems*, *120*, 106 – 117.

Lee, D. J., & Thornton, A. C. (1996). The identification and use of key characteristics in the product development process. volume Volume 4: 8th International Conference on Design Theory and Methodology of *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*.

Li, A.-D., He, Z., Wang, Q., & Zhang, Y. (2019). Key quality characteristics selection for imbalanced production data using a two-phase bi-objective feature selection method. *European Journal of Operational Research*, *274*(3), 978 – 989.

Li, A.-D., He, Z., & Zhang, Y. (2016). Bi-objective variable selection for key quality characteristics selection based on a modified NSGA-II and the ideal point method. *Computers in Industry*, *82*, 95 – 103.

Li, A.-D., He, Z., & Zhang, Y. (2022). Robust multi-response optimization considering location effect, dispersion effect, and model uncertainty using hybridization of nsga-ii and direct multi-search. *Computers & Industrial Engineering*, *169*, 108247.

Li, A.-D., Xue, B., & Zhang, M. (2020). Multi-objective feature selection using hybridization of a genetic algorithm and direct multisearch for key quality characteristic selection. *Information Sciences*, *523*, 245 – 265.

Li, A.-D., Xue, B., & Zhang, M. (2021). Improved binary particle swarm optimization for feature selection with new initialization and search space reduction strategies. *Applied Soft Computing*, *106*, 107302.

Li, S., & Chen, Y. (2016). A bayesian variable selection method for joint diagnosis of manufacturing process and sensor faults. *IIE Transactions*, *48*(4), 313–323. doi:10.1080/0740817X.2015.1109739.

Liu, S., Wang, H., Peng, W., & Yao, W. (2022). A surrogate-assisted evolutionary feature selection algorithm with parallel random grouping for high-dimensional classification. *IEEE Transactions on Evolutionary Computation*, (pp. 1–1). doi:10.1109/TEVC.2022.3149601.

Nguyen, B. H., Xue, B., Andreae, P., Ishibuchi, H., & Zhang, M. (2020). Multiple reference points-based decomposition for multiobjective feature selection in classification: Static and dynamic mechanisms. *IEEE Transactions on Evolutionary Computation*, *24*(1), 170–184.

Nguyen, B. H., Xue, B., Andreae, P., & Zhang, M. (2021). A new binary particle swarm optimization approach: Momentum and dynamic balance between exploration and exploitation. *IEEE Transactions on Cybernetics*, *51*(2), 589–603.

Nguyen, H. B., Xue, B., Liu, I., Andreae, P., & Zhang, M. (2016). New mechanism for archive maintenance in PSO-based multi-objective feature selection. *Soft Computing*, *20*(10), 3927–3946.

Oh, I., Lee, J., & Moon, B. R. (2004). Hybrid genetic algorithms for feature selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *26*(11), 1424–1437.

Pacheco, J. A., Casado, S., Ángel-Bello, F., & Alvarez, A. M. (2013). Bi-objective feature selection for discriminant analysis in two-class classification. *Knowledge-Based Systems*, *44*, 57 – 64.

Robnik-Sikonja, M., & Kononenko, I. (2003). Theoretical and empirical analysis of relieff and rrelieff. *Mach. Learn.*, *53*(1-2), 23–69.

Rodriguez, J. D., Perez, A., & Lozano, J. A. (2010). Sensitivity analysis of k-fold cross validation in prediction error estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *32*(3), 569–575.

Shi, J., & Zhou, S. (2009). Quality control and improvement for multistage systems: A survey. *IIE Transactions*, *41*(9), 744–753.

Sugiyama, M. (2007). Dimensionality reduction of multimodal labeled data by local fisher discriminant analysis. *Journal of Machine Learning Research*, *8*, 1027–1061.

Tomczyk, M. K., & Kadziński, M. (2020). Decomposition-based interactive evolutionary algorithm for multiple objective optimization. *IEEE Transactions on Evolutionary Computation*, *24*(2), 320–334.

Tsung, F., Li, Y., & Jin, M. (2008). Statistical process control for multistage manufacturing and service operations: a review and some extensions. *International Journal of Services Operations and Informatics*, *3*(2), 191–204.

Wan, Y., Wang, M., Ye, Z., & Lai, X. (2016). A feature selection method based on modified binary coded ant colony optimization algorithm. *Applied Soft Computing*, *49*, 248–258.

Wang, A., & Shi, J. (2021). Holistic modeling and analysis of multistage manufacturing processes with sparse effective inputs and mixed profile outputs. *IISE Transactions*, *53*(5), 582–596.

Wang, J., Zheng, P., & Zhang, J. (2020). Big data analytics for cycle time related feature selection in the semiconductor wafer fabrication system. *Computers & Industrial Engineering*, *143*, 106362.

Wang, P., Xue, B., Liang, J., & Zhang, M. (2021). Multiobjective differential evolution for feature selection in classification. *IEEE Transactions on Cybernetics*, (pp. 1–15). doi:10.1109/TCYB.2021.3128540.

Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics Bulletin*, *1*(6), 80 – 83.

Wold, S., Sjöström, M., & Eriksson, L. (2001). PLS-regression: a basic tool of chemometrics. *Chemometrics and Intelligent Laboratory Systems*, *58*(2), 109 – 130.

Xue, B., Zhang, M., & Browne, W. N. (2013). Particle swarm optimization for feature selection in classification: A multi-objective approach. *IEEE Transactions on Cybernetics*, *43*(6), 1656–1671.

Xue, B., Zhang, M., Browne, W. N., & Yao, X. (2016). A survey on evolutionary computation approaches to feature selection. *IEEE Transactions on Evolutionary Computation*, *20*(4), 606–626.

Yang, H., Kumara, S., Bukkapatnam, S. T., & Tsung, F. (2019). The internet of things for smart manufacturing: A review. *IISE Transactions*, *51*(11), 1190–1216.

Yu, L., & Liu, H. (2004). Efficient feature selection via analysis of relevance and redundancy. *Journal of Machine Learning Research*, *5*(Oct.), 1205–1224.

Yuan, Y., Xu, H., Wang, B., & Yao, X. (2016). A new dominance relation-based evolutionary algorithm for many-objective optimization. *IEEE Transactions on Evolutionary Computation*, *20*(1), 16–37.

Zhang, Q., & Li, H. (2007). MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, *11*(6), 712–731.

Zhang, Y., Gong, D., Gao, X., Tian, T., & Sun, X. (2020). Binary differential evolution with self-learning for multi-objective feature selection. *Information Sciences*, *507*, 67–85.

Zhang, Y., Wang, Y.-H., Gong, D.-W., & Sun, X.-Y. (2022). Clustering-guided particle swarm feature selection algorithm for high-dimensional imbalanced data with missing values. *IEEE Transactions on Evolutionary Computation*, *26*(4), 616–630.

Zhou, Y., Kang, J., Kwong, S., Wang, X., & Zhang, Q. (2021). An evolutionary multi-objective optimization framework of discretization-based feature selection for classification. *Swarm and Evolutionary Computation*, *60*, 100770.

Zitzler, E., & Thiele, L. (1999). Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, *3*(4), 257–271.