MATEMATIKA, MJIM, Volume 38, Number 3, 195–207

Surface Mount Technology Line Optimisation using Modified *k*-means with Feature Weight Constraints

Xian Xiang Wong, Wen Eng Ong and Amirah Rahman*
School of Mathematical Sciences, Universiti Sains Malaysia
11800 Penang, Malaysia.
*Corresponding author: amirahr@usm.my

Article history
Received: 27 October 2022
Received in revised form: 6 December 2022
Accepted: 15 December 2022
Published online: 31 December 2022

Abstract The common setup problem for chip mounters in a Surface Mount Technology (SMT) line is to group different device models, each of them consisting of different components, into minimum number of clusters where each cluster has a maximum component size. This type of setup problem is classified as a clustering problem with feature weight constraints which cannot be fulfilled by traditional data clustering. In this study, we introduce some vital modifications on standard *k*-means algorithm such that it can incorporate feature weight constraints adapted from cluster size constraints. We also propose a modification to the elbow method to determine the number of clusters of the clustering problem with feature weight constraints. The results show that the proposed algorithm (modified *k*-means with feature weight constraints) is able to fulfill the feature weight constraints and solve the common setup problem. The results verify the proposed algorithm has superior performance over standard *k*-means algorithm in clustering problem with feature weight constraints. For the common setup problem on a given data set, the analysis shows that 1000 runs of the proposed algorithm implemented using MATLAB is able to obtain at least one valid clustering result within a reasonable run time.

Keywords Clustering; Modified *k*-means Algorithm; Feature Weight Constraints; Elbow Method; Production Line Optimisation

Mathematics Subject Classification 90C59

1 Introduction

Surface Mount Technology (SMT) is a method for producing electronic circuits in which the components are mounted or placed directly onto the surface of printed circuit boards (PCBs) or substrates [1]. PCB is a substrate of epoxy glass, clad metal, or other material upon which a pattern of conductive traces is formed to interconnect components [1]. Examples of components include resistors, capacitors, and IC chips. An electronic device soldered to a PCB is called surface mount device. SMT components usually require less space which allows components to be mounted more compactly onto the PCB. Hence, newer devices have more functions, perform faster and are more

portable at the same time. An SMT process called chip mounting mounts or places components directly onto the surface of PCB before soldering process. The machine used in chip mounting process is known as chip mounter.

1.1 Common Setup Problem

Different types of device models consist of different types and different numbers of components and have different configurations. A simple illustration in Figure 1(a) shows the differences in both types and configurations of components in 3 different device models. Note that C1 to C7 are components. Device model A consists of components C1, C2 and C3, device model B consists of C4, C5, C6 and C7 and device model C consists of C3, C4, C6 and C7. If we want to produce device model B after device model A, we need to change the setup of chip mounters. This process is called conversion. The time taken for a conversion is an average of 30 mins. The conversion time is significant if the number of conversions is high and thus reduces the productivity.

It makes more sense to group device models that share more common components. In Figure 1(a), device models B and C use 3 same components (C4, C6 and C7). Thus, it is better to group device model B and C in the same sub-setup as they are more "similar" to one another compared to device model A. Then, the common setup for device models B and C has 5 components (C3, C4, C5, C6 and C7). Hence, the common setup problem is to group those similar device models together into a common setup, which is a clustering problem.

Two chip mounters are used in a production line as shown in Figure 1(b). Thus, one common setup consists of two sub-setups (feeder tables). Each feeder table has 34 component vacancies and each component takes 1 or 2 component vacancies (or component size) as illustrated in Figure 2. Hence, we know that the common setup problem is a clustering problem with some constraints.

Given n distinct device models with m distinct components where each component takes 1 or 2 component vacancies, we need to partition n device models into minimum k disjoint sub-setups without violating the maximum number of component vacancies in each sub-setup. Then pair up the sub-setups into groups of two to obtain an optimal list of $\lceil k/2 \rceil$ common setups for the production line of chip mounting.

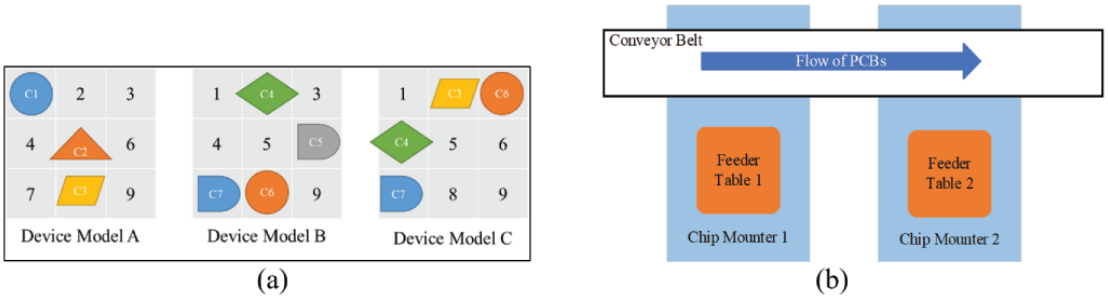

Figure 1: (a) Example of Device Models (b) Plan View of Chip Mounter Production Line

2 Cluster Analysis

Cluster analysis is defined as grouping of similar objects [2], a sub-field in data mining that specializes in techniques for finding similar groups in a large database [3], or an effort to classify similar objects

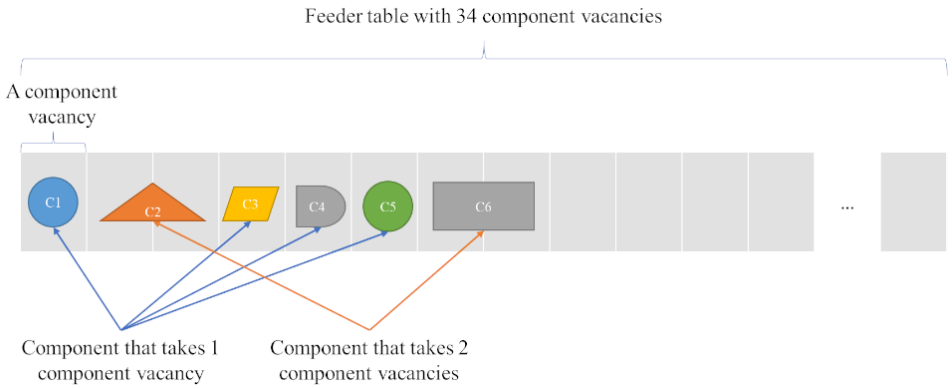

Figure 2: Table Filled with Components

in the same groups [4]. Clustering groups data objects together based on the information found in the objects themselves which describe the data objects and their relationship [5]. The objective of a clustering method is to construct high quality clusters with high degree of intra-cluster similarity and low degree of inter-cluster similarity [4]. In other words, a good clustering result will have the members in same cluster have high degree of similarity to one another and low degree of similarity with the members of other clusters.

The similarity between two objects is normally quantified with a distance measure in clustering [6]. There are numerous kinds of distance measures, and the results of clustering algorithms may depend on the distance measure used. The distance between points $\mathbf{x}$ and $\mathbf{y}$ is denoted as $d(\mathbf{x}, \mathbf{y})$ where $\mathbf{x} = (x_1, x_2, \cdots, x_m)$ and $\mathbf{y} = (y_1, y_2, \cdots, y_m)$ are two points in dimension m. The common distance measures used in clustering algorithms are Euclidean, squared Euclidean, Manhattan and Minkowski.

t A cluster is normally represented by a central point called centroid [5]. The objects within the cluster is closer (more similar) to its centroid than the centroids of other clusters. The centroid, $\boldsymbol{\mu}_j$ is defined as the average of all points (or objects) in the cluster, C_j as follows [5][7]:

$$\boldsymbol{\mu}_j = \frac{1}{|C_j|} \sum_{\mathbf{x}_i \in C_j} \mathbf{x}_i . \# (1)$$

Cluster analysis is an important analysis tool in many fields such as big data clustering [8], document clustering [9], image segmentation [10] [11] and information retrieval [12]. The major clustering methods can be generally classified into five categories: partitioning, hierarchical, grid-based, model-based and density-based [13][14].

2.1 *k*-means Clustering

One of the most well-known partitional clustering methods is $k-$means due to its simplicity in [4][7]. The ability of $k-$means method to cluster huge data quickly and efficiently made it a very popular clustering method. However, $k-$means algorithm is very sensitive to initial centroids. Due to random initialization of the initial centroids, $k-$means does not guarantee a unique clustering result [15]. When random initialization of centroids is used, different runs of $k-$means will give different sum of squared error (SSE) values [5]. When random initial centroids close to the optimum solution, $k-$means has high possibility to find out the optimum clustering result. Otherwise, it often led

to poor clustering results [16]. Thus, despite of the efficiency of $k-$means it often produces a suboptimal clustering result that is only a local minimum instead of global minimum, mainly due to the randomness in its initialization [17]. One commonly used technique used to address the problem of choosing initial centroids is to perform multiple runs, each with different set of randomly chosen initial centroids, then choose the clustering result with the least SSE.

Another challenge is to determine the number of clusters, k [18]. This is because $k-$means requires a predefined k value but it is not always known. Elbow method is an unsupervised cluster evaluation measure that helps determine the k value. The elbow method is a visual method that involves plotting a line chart of the SSE values against the number of clusters. If the line chart resembles an arm, then the "elbow" (the point of inflection of the curve) is a good indication that the underlying model fits best at the point [5][18].

2.2 Common Setup Problem as a Clustering Problem with Constraint

The common setup problem is a clustering problem which involves constraints on cluster's total feature weight. Feature weights refer to the weights of the objects being clustered. In standard k-means, all features have the same weight and there are no constraints on feature weight. To our knowledge, there is no study of $k-$means algorithm that can incorporate feature weights and constraint on size for each cluster. Therefore, this study proposes a modification to the standard $k-$means algorithm where it can incorporate the feature weight constraints for each cluster.

3 Methodology

Given n objects (device models) with m features (components) where each component takes either 1 or 2 feature weights (component vacancies), we need to partition n objects into minimum k disjoint clusters (sub-setups) without violating feature weight constraints (maximum number of component vacancies) in each cluster. Then pair up the clusters into groups of two to obtain an optimal list of $\left\lceil \frac{k}{2} \right\rceil$ common setups for the chip mounting production line.

3.1 Problem Formulation

We define the mathematical terms formally. Let $F = \{f_1, f_2, \cdots, f_m\}$ be a given data set of m features. Let the feature weight vector be $\boldsymbol{w} = (w_1, w_2, \cdots, w_m) \in \mathbb{R}^m$ where w_j is the feature weight of f_j, $\forall j \in \{1, 2, \cdots, m\}$. Let $X = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$ be a given set of n objects where $\mathbf{x}_i = (x_{i1}, x_{i2}, \cdots, x_{im}) \in \mathbb{R}^m$ such that $\forall\, i \in \{1, 2, \cdots, n\}$, $j \in \{1, 2, \cdots, m\}$, the feature value,

$$x_{ij} = \begin{cases} 1, \text{ if } \mathbf{x}_i \text{ consists of } f_j \\ 0, \text{ otherwise} \end{cases}.$$

In a clustering problem, the objective of a clustering algorithm is to find k clusters, $C = \{C_1, C_2, \cdots, C_k\}$ where $1 \leq k \leq n$.

Next, we define some mathematical operations that will be used in the algorithms formally. We define the *vector addition*, '+' for $\mathbb{R}^m$ component-wise, using the addition defined on $\mathbb{R}$. If $\mathbf{x}_a = (x_{a1}, x_{a2}, \cdots, x_{am}) \in \mathbb{R}^m$ and $\mathbf{x}_b = (x_{b1}, x_{b2}, \cdots, x_{bm}) \in \mathbb{R}^m$, then

$$\mathbf{x}_a + \mathbf{x}_b = (x_{a1} + x_{b1}, x_{a2} + x_{b2}, \cdots, x_{am} + x_{bm}) \in \mathbb{R}^m.$$

We define the *scalar multiplication* for $\mathbb{R}^m$ component-wise. If $\mathbf{x}_a=(x_{a1}, x_{a2}, \cdots, x_{am}) \in \mathbb{R}^m$ and $\lambda \in \mathbb{R}$, then

$$\lambda \mathbf{x}_a=(\lambda x_{a1}, \lambda x_{a2}, \cdots, \lambda x_{am}) \in \mathbb{R}^m.$$

We define the *inclusion disjunction*, '$\vee$' for $\mathbb{R}^m$ component-wise, using the inclusion disjunction defined on $\mathbb{R}$. If $\mathbf{x}_a=(x_{a1}, x_{a2}, \cdots, x_{am})$ and $\mathbf{x}_b=(x_{b1}, x_{b2}, \cdots, x_{bm}) \in \mathbb{R}^m$, then $\mathbf{x}_a \vee \mathbf{x}_b=(x_{a1} \vee x_{b1}, x_{a2} \vee x_{b2}, \cdots, x_{am} \vee x_{bm}) \in \mathbb{R}^m$, where $(x_{ac} \vee x_{bc}) \in \{0, 1\}$, $\forall c \in \{1, 2, \cdots, m\}$. Note that

$$\bigvee_{i=1}^{n} \mathbf{x}_i = \mathbf{x}_1 \vee \mathbf{x}_2 \vee \cdots \vee \mathbf{x}_n.$$

We define the *dot product*, '$\cdot$' for $\mathbb{R}^m$; i.e., if $\mathbf{w}=(w_1, w_2, \cdots, w_m) \in \mathbb{R}^m$ and $\mathbf{x}_i=(x_{i1}, x_{i2}, \cdots, x_{im}) \in \mathbb{R}^m$, then

$$\mathbf{w} \cdot \mathbf{x}_i = w_1 x_{i1} + w_2 x_{i2} + \cdots + w_m x_{im} = \sum_{a=1}^{m} w_a x_{ia} \in \mathbb{R}.$$

Lastly, the feature weight of each cluster C_j is defined as

$$wt\left(C_j\right) = \mathbf{w} \cdot \left(\bigvee_{\mathbf{x}_i \in C_j} \mathbf{x}_i \right),$$

such that $1 \leq j \leq k$. In the data clustering with feature weight constraints, the feature weight constraints, ω_j is available for each cluster C_j such that $1 \leq j \leq k$. Therefore, a feature weight constrained data clustering algorithm must satisfy an extra constraint:

$$wt\left(C_j\right) \leq \omega_j, \forall j \in \{1, 2, \cdots, k\}.$$

3.2 Modified $k-$means with Feature Weight Constraints

A modified $k-$means algorithm is proposed here for data clustering with feature weight constraints. The standard $k-$means algorithms is as shown in Figure 3.

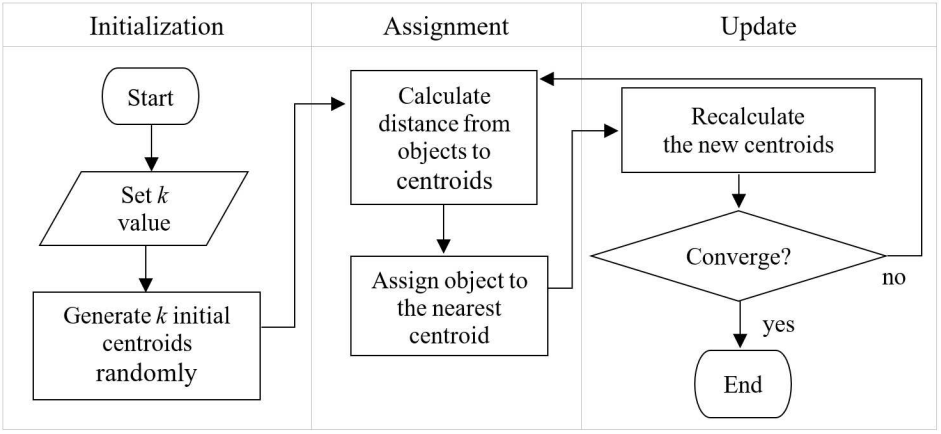

Figure 3: Standard k-means Flow Chart

The proposed solution method can be categorized into 4 steps which are *Initialization step, Assignment step, Update step* and *Validation step* as shown in Figure 4.

The modified $k-$means algorithm is adapted from the standard $k-$means algorithm with two changes. The Assignment step is modified to handle feature weight constraints [7]. Validation step is added to handle the situation of clustering results with outliers. We introduce two objective functions for modified $k-$means with feature weight constraints. In the common setup problem, we know that the fewer component vacancies used, the lesser the work needed to install component reels in the chip mounter. Thus, the first objective function is to minimize the total feature weight (which is the total number of component vacancies used),

$$z_1 = \sum_{j=1}^{k} wt\left(C_j\right) .\# \,(2)$$

We may obtain *outliers* (objects failed to be assigned into any cluster) after the modified $k-$means algorithm terminates. Thus, we introduce *penalty value*, ρ into the SSE for each outlier found in the clustering result to get

$$z_2 = \left(\sum_{j=1}^{k} \sum_{\mathbf{x}_i \in C_j} d\left(\mathbf{x}_i, \boldsymbol{\mu}_j\right)\right) + \left|C_q\right| \rho \# \,(3)$$

where $\left|C_q\right|$ is the number of outliers (outlier cluster size) and each ρ added increases z_2 value significantly ($\rho \gg z$). Thus, our second objective function (or *validation criterion*) is $z_2 < \rho$ to ensure the clustering result has no outliers.

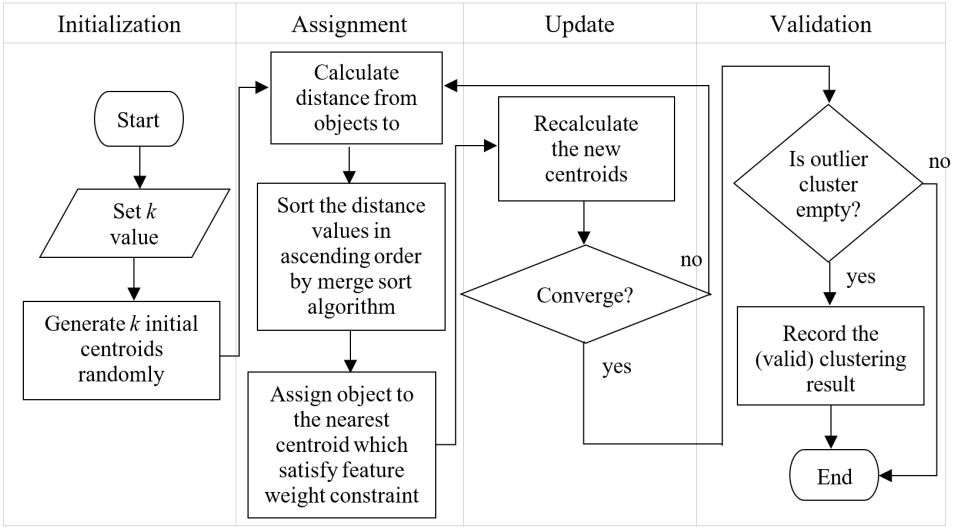

Figure 4: Modified $k-$means with Feature Weight Constraints Flow Chart

3.2.1 Initialization Step

Define the value of k and randomly generate initial centroids $\boldsymbol{\mu}_j^{(1)}$ of cluster C_j.

3.2.2 Assignment Step

Assign each object to the closest centroid only if the feature weight of the cluster does not reach its maximum capacity, $wt\left(C_j^{(t)}\right) \leq \omega_j$. However, a situation may arise where an object is not able to be

assigned into any clusters as all clusters have reached their maximum feature weights. Then the object is assigned to the outlier cluster, $C_q^{(t)}$ (while implementing, we can simply set $q = -1$ or $q > k$). The outlier cluster is used to record the outliers, which acts as an indicator for invalid clustering result.

While implementing the assignment step, this can be easily achieved by sorting the values of square Euclidean distance, $d\left(\boldsymbol{x}_i, \boldsymbol{\mu}_j^{(t)}\right)$ in ascending order for all $i \in \{1, 2, \cdots, n\}$, $j \in \{1, 2, \cdots, k\}$ and iterating through the sorted array until it finds a cluster which satisfies the feature weight constraints,

$$wt\left(C_j^{(t)} \cup \{\boldsymbol{x}_i\}\right) = \boldsymbol{w} \cdot \left(\left(\bigvee_{\boldsymbol{x}_p \in C_j^{(t)}} \boldsymbol{x}_p\right) \vee \boldsymbol{x}_i\right) \leq \omega_j.$$

In this case, we use merge sort algorithm [19] to sort $d\left(\boldsymbol{x}_i, \boldsymbol{\mu}_j^{(t)}\right)$ in ascending order. As per the standard $k-$means algorithm, each object is assigned to only one cluster in each iteration.

Suppose we have k centroids and n objects. Thus, we will have $k \times n$ squared Euclidean distances between object i and centroid j, $d\left(\boldsymbol{x}_i, \boldsymbol{\mu}_j^{(t)}\right)$ where $i \in \{1, 2, \cdots, n\}$, $j \in \{1, 2, \cdots, k\}$. Using merge sort algorithm, we sort the list of distances, $d\left(\boldsymbol{x}_i, \boldsymbol{\mu}_j^{(t)}\right)$ in ascending order. Then we assign all objects to the nearest cluster provided the feature weight constraint must be satisfied. We start to assign from the least $d\left(\boldsymbol{x}_i, \boldsymbol{\mu}_j^{(t)}\right)$ value. For each assignation, we have several cases:

Case 1: The object i is not in any cluster yet.

Case 1.1 The object i fulfills the feature weight constraint, ω_j.

This means $wt\left(C_j^{(t)} \cup \{\boldsymbol{x}_i\}\right) = \boldsymbol{w} \cdot \left(\left(\bigvee_{\boldsymbol{x}_p \in C_j^{(t)}} \boldsymbol{x}_p\right) \vee \boldsymbol{x}_i\right) \leq \omega_j$ is satisfied. Object i is assigned to cluster j.

The object i does not fulfill the feature weight constraint, ω_j. Which means $wt\left(C_j^{(t)} \cup \{\boldsymbol{x}_i\}\right) = \boldsymbol{w} \cdot \left(\left(\bigvee_{\boldsymbol{x}_p \in C_j^{(t)}} \boldsymbol{x}_p\right) \vee \boldsymbol{x}_i\right) \leq \omega_j$ is not satisfied. Then the object i will not be assigned into cluster j. If object i fails to be assigned to any cluster, we will assign object i to outlier cluster, $C_q^{(t)}$.

Case 1.2: The object i is in one of the clusters.

Since the object i is in one of the clusters and we know that each object only can be in exactly one cluster. Thus, the object i will not be assigned into cluster j.

After all assignations, there are 3 possible situations:

Situation 1: The outlier cluster is empty, and each cluster has at least one object.

- The clustering result is valid. This is the ideal situation and we will record this result.

Situation 2: The outlier cluster is empty, but there exists at least one empty cluster.

- The clustering result is valid. We may record this result, but we should use lower k value.

Situation 3: The outlier cluster is nonempty.

- The clustering result is invalid. This means that all k clusters have reached their maximum feature weights. We will not record this result as we do not want to have any outliers.

3.2.3 Update Step

Recompute the centroids of each cluster:

$$\boldsymbol{\mu}_j^{(t+1)} = \frac{1}{\left|C_j^{(t)}\right|} \sum_{\mathbf{x}_i \in C_j^{(t)}} \mathbf{x}_i.$$

3.2.4 Validation Step

After the convergence of algorithm, there are 3 possible situations:

- Situation 1: The outlier cluster is empty, and each cluster has at least one object.
- Situation 2: The outlier cluster is empty, but there exists at least one empty cluster.
- Situation 3: The outlier cluster is nonempty.

In *validation step*, we record the results for Situation 1 and 2 only.
While implementing this algorithm, we compute z_2 (SSE with penalty value) from equation (**??**) and use the validation criterion as $z_2 < \rho$. This means that if at least one outlier is found in the outlier cluster, z_2 value will surely exceed the validation criterion.

3.3 Determine the Best Clustering Result

Since this modified $k-$means uses random initialization of initial centroids, the clustering result for each run is different. Thus, we perform multiple runs, each with a different set of randomly chosen initial centroids. Then we choose the clustering result with the lowest total feature weight, z_1 which fulfills the validation criterion $z_2 < \rho$.
As per standard $k-$means, modified $k-$means requires a predefined k value during execution. Thus, we introduce a modification to the elbow method to determine the number of clusters (k value) of the clustering problem with feature weight constants. The modification is in terms of adding a penalty value ρ, which serves as an upper limit for z_2.
The steps of the modified elbow method are as follows:

i. Set penalty value ρ.

ii. For each k where $1 \leq k \leq n$, perform multiple runs of the modified algorithm and record the least z_2 value.

iii. Plot the line chart of z_2 values against k values.

iv. Plot a horizontal benchmark, $z_2 = \rho$ on the line chart.

v. Then the first k value below the benchmark will be the least k value which does not produce outliers.

4 Application, Result and Analysis

4.1 Solving the SMT Common Setup Problem

The data set of the common setup problem as follows:

i. A list of 44 distinct device models with total of 123 distinct components.

ii. The component size of the 123 components.

iii. The maximum number of component vacancies in each sub-setup is 34.

4.1.1 Formulating the SMT Common Setup Problem

We know that $n = 44$ and $m = 123$. Then we formulate the data of SMT common setup problem into mathematical form. Thus, we have:

- the data set of 44 objects is denoted as $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_{44}\}$,
- the data set of 123 features is denoted as $F = \{f_1, f_2, \cdots, f_{123}\}$,
- the feature weight constraints of cluster C_j is $\omega_j = 34$, $\forall j \in \{1,\ 2, \cdots, k\}$.
- the feature weight vector respect to data set F is denoted as $\boldsymbol{w} = (w_1, w_2, \cdots, w_{123})$.

Then we need to find a list of k clusters which are $C = \{C_1, C_2, \cdots, C_k\}$.

4.1.2 Determine k Value Using Modified Elbow Method

We perform modified elbow method to determine the number of clusters, k. For this data set, we set the penalty value $\rho = 999$, which is determined by the pre-knowledge of the range of z_2 values of valid clustering results. We perform modified $k-$means with feature weight constraints for 1000 times to obtain the least z_2 values for each k where $k \in \{1,\ 2,\ \cdots\ ,\ 10\}$. The least z_2 and the number of outliers for each number of clusters, k is shown in Table 1. From the modified elbow method plot shown in Figure 5, we know that the recommended k value is 6.

Table 1: The z_2 Values and The Number of Outliers Based on Different k Values

Number of clusters, k	z_2	Number of outliers
1	33016.636	33
2	15147.928	15
3	8187.963	8
4	4192.012	4
5	2194.012	2
6	203.638	0
7	155.667	0
8	127.833	0
9	101.545	0
10	87.617	0

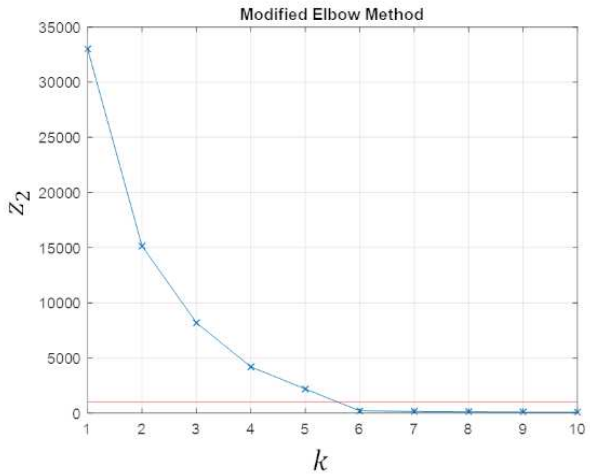

Figure 5: Modified Elbow Method for $k \in \{1,\ 2,\ \cdots,\ 10\}$

4.1.3 Results

We implement our proposed algorithm using MATLAB on the data set. By setting $k = 6$, we obtain a valid clustering result with 6 clusters (sub-setups) as shown in Table 2. All clusters satisfy the feature weight constraint:

$$wt\left(C_j\right) \leq 34 = \omega_j,\ \forall j \in \{1,\ 2,\ 3,\ 4,\ 5,\ 6\}.$$

By pairing up all clusters into groups of 2, we obtain $\left\lceil \frac{k}{2} \right\rceil = \left\lceil \frac{6}{2} \right\rceil = 3$ common setups for chip mounting. Thus, this shows that modified $k-$means with feature weight constraints can solve the common setup problem.

4.2 Analysis

4.2.1 Performance Analysis of Modified $k-$means Algorithm

We evaluate and analyse the performance of the modified $k-$means algorithm against the standard $k-$means algorithm using the data set of the common setup problem. Both algorithms were implemented using MATLAB with $k = 6$. The clustering results of these two algorithms are summarized in Table 3.

From Table 3, the clustering result of $k-$means violates the feature weight constraint of cluster 1, ω_1 $(78 > 34)$. Thus, standard $k-$means fails to cluster the data set of common setup problem, which means it fails to solve the clustering problem with feature weight constraints. In the case of modified $k-$means, the clustering results fulfils all the feature weight constraints. Thus, the modified $k-$means algorithm can cluster the data set of common setup problem. Therefore, this result verifies the superior performance of the proposed algorithm over the standard $k-$means in dealing with clustering problem with feature weight constraints.

Table 2: The Clustering Result of Common Setup in 6 Sub-setups

Cluster	1	2	3	4	5	6
Cluster Size	2	16	6	8	2	10
Feature Weight	22	31	34	33	24	32
Device Model	D26	D01	D10	D23	D42	D11
	D41	D02	D20	D24	D43	D12
		D03	D21	D25		D13
		D04	D22	D32		D14
		D05	D28	D38		D15
		D06	D29	D39		D16
		D07		D40		D30
		D08		D44		D31
		D09				D33
		D17				D34
		D18				
		D19				
		D27				
		D35				
		D36				
		D37				

Table 3: Comparison of Clustering Results

Cluster index, j	Feature weight constraints, ω_j	Final feature weight	
		Standard $k-$means	Standard $k-$means
1	34	78	22
2	34	15	31
3	34	22	34
4	34	24	33
5	34	23	24
6	34	17	32

4.2.2 Run Time Analysis of Modified $k-$means with Feature Weight Constraints

While implementing modified $k-$means algorithm with feature weight constraints, we might obtain an invalid clustering result if we do not perform sufficient number of runs. Thus, we will always perform multiple runs of the algorithm and choose the best among the valid clustering results. We know that the higher the number of runs, the higher the chances to obtain the best valid clustering result. However, the cost of the large number of runs might be expensive (longer run time). Hence, we need to find a balance between number of runs and run time and still be able to obtain a valid clustering result.

Using the same data set with $k = 6$, we perform different number of runs of the algorithm and obtain the run time, number of valid clustering results and the least total feature weight (total feature

weight of the best valid clustering result), as shown in Table 4. We choose the number of runs to be 1, 100, 1000 and 10,000 based on preliminary experimentation.
Table 4: Comparison of Different Number of Runs

Table 4: Comparison of Different Number of Runs

Number of runs	1	100	1000	10000
Run Time (s)	0.008266	0. 739307	5.326081	55.176309
Number of valid clustering results	0	2	22	133
Least total feature weigh	N/A	183	176	176

The results from Table 4 are not unique. When we perform 10000 runs of the algorithm, only 133 out of 10000 clustering results are valid. Thus, the percentage of the valid results is approximately 1.33%. The least total feature weight obtained is 176. However, the run time for 10000 runs is 55.2 seconds which is not practical. Thus, we should use fewer runs while still obtaining valid clustering solution. In this case, it is reasonable to choose the number of runs as 1000.

5 Conclusion

The common setup problem in SMT line is a clustering problem with feature weight constraints that cannot be solved by implementing traditional data clustering method. In this study, we introduce modifications on the $k-$means algorithm to incorporate feature weight constraints. The *Assignment step* is adapted and modified from the *Assignment step* of the $k-$means algorithm with cluster size constraints to account for the feature weight constraints of each cluster. In the *Validation step*, we add penalty value to SSE for each outlier found in the outlier cluster. If at least one outlier is found in the outlier cluster, SSE will surely exceed the *validation criterion*, thus only the clustering results without outliers will be recorded. We also modified the elbow method with penalty value to determine the number of clusters.

The results verify that the proposed algorithm has superior performance over standard $k-$means algorithm in clustering problem with feature weight constraints. For the common setup problem, the analysis shows that 1000 runs of the proposed algorithm implemented using MATLAB is able to obtain at least one valid clustering result within a reasonable run time. By using the proposed algorithm, a minimum of 3 common setups (or 6 sub-setups) were obtained.

One limitation of this study is that the initial centroids are randomly generated (as per standard $k-$means). Hence, future studies could explore methods that are able to find better initial centroids. Future studies could also include more constraints such as for each common setup, one of the two sub-setups must only consist of components with size 1. Besides that, one may explore other clustering methods to incorporate feature weight constraints.

References

[1] 1 Prasad, R. *Surface mount technology: principles and practice.* Springer Science & Business Media. 2013.

[2] 2 Hartigan, J. A. *Clustering Algorithms.* New York: John Wiley & Sons, Inc. 1975.

[3] 3 Nanopoulos, A., Theodoridis, Y. and Manolopoulos, Y. C2P: Clustering based on closest pairs. 2001. In *VLDB* 331-340.

[4] 4 Ezugwu, A. E., Ikotun, A. M., Oyelade, O. O., Abualigah, L., Agushaka, J. O., Eke, C. I. and Akinyelu, A. A. A comprehensive survey of clustering algorithms: State-of-the-art machine learning applications, taxonomy, challenges, and future research prospects, *Engineering Applications of Artificial Intelligence.* 2022. 110: 104743.

[5] 5 Tan, P. N., Steinbach, M. and Kumar, V. Cluster analysis: basic concepts and algorithms, *Introduction to Data Mining.* Boston: Addison-Wesley. 2005. 487-568.

[6] 6 de Amorim, R. C. Learning feature weights for K-Means clustering using the Minkowski metric, *Department of Computer Science and Information Systems Birkbeck, University of London.* 2011.

[7] 7 Ganganath, N., Cheng, C. T. and Chi, K. T. Data clustering with cluster size constraints using a modified k-means algorithm. *2014 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery.* 2014. 158-161.

[8] 8 Nasraoui, O. and N'Cir, C. E. B. Clustering methods for big data analytics. *Techniques, Toolboxes and Applications.* 2019. 1: 91-113.

[9] 9 Curiskis, S. A., Drake, B., Osborn, T. R. and Kennedy, P. J. An evaluation of document clustering and topic modelling in two online social networks: Twitter and Reddit, *Information Processing & Management.* 2020. 57(2): 102034.

[10] 10 Hrosik, R. C., Tuba, E., Dolicanin, E., Jovanovic, R. and Tuba, M. Brain image segmentation based on firefly algorithm combined with k-means clustering," *Stud. Inform. Control.* 2019. 28(2): 167-176.

[11] 11 Lei, T., Zhang, X., Jia, X. H., Liu, S. G. and Zhang, Y. N. Research progress on image segmentation based on fuzzy clustering. *Acta Electonica Sinica.* 2019. 47(8): 1776.

[12] 12 Bellot, P. and El-Bèze, M. A clustering method for information retrieval. *Technical ReportIR-0199, Laboratoire d'Informatique d'Avignon, France*, 1999.

[13] 13 Mazarbhuiya, F. A., Alzahrani, M. Y. and Mahanta, A. K. Detecting Anomaly Using Partitioning Clustering with Merging. *ICIC Express Letters.* 2020. 14(10): 951-960.

[14] 14 Patel, D., Modi, R. and Sarvakar, K. A Comparative Study of Clustering Data Mining: Techniques and Research Challenges. *International Journal of Latest Technology in Engineering, Management & Applied Science.* 2014, 3(9): 67-70.

[15] 15 Khan, S. S. and Ahmad, A. Cluster center initialization algorithm for K-means clustering. *Pattern recognition letters.* 2004. 25(11): 1293-302.

[16] 16 Cheung, Y. M. k∗-Means: A new generalized k-means clustering algorithm. *Pattern Recognition Letters.* 2003. 24(15): 2883-93.

[17] 17 Kövesi, B., Boucher, J. M. and Saoudi, S. Stochastic K-means algorithm for vector quantization. *Pattern Recognition Letters.* 2001. 22(6-7): 603-10.

[18] 18 Kodinariya, T. M. and Makwana, P. R. Review on determining number of Cluster in K-Means Clustering. *International Journal of Advance Research in Computer Science and Management Studies.* 2013. 1(6): 90-5.

[19] 19 Knuth, D. E. Sorting by merging. The Art of Computer Programming. *Sorting and Searching.* 1998. 158-168.