Available online at www.sciencedirect.com
ScienceDirect
Expert Systems with Applications 33 (2007) 324–329
Expert Systems with Applications
www.elsevier.com/locate/eswa

Visualization of machine clustering for a Taiwanese IC packaging foundry

Hsu-Hao Yang [a,*], He-Yau Kang [b], Tzu-Chiang Liu [b]

[a] *Institute of Production Systems Engineering and Management, National Chinyi Institute of Technology, 35, Lane 215, Chungshan Road, Taiping 411, Taiwan*
[b] *Department of Industrial Engineering and Management, National Chinyi Institute of Technology, Taiping 411, Taiwan*

Abstract

This paper applies a two-phase methodology to cluster 366 records of the wire bond machines for a Taiwanese IC packaging foundry, where six attributes of each machine are chosen to cluster. The purpose of clustering in this paper is to help the foundry more effectively assign a family of machines to orders that appear in various forms such as emergent or quality-demanding and are highly dynamic in nature. Given the clusters, we use the technique of parallel coordinates to plot each attribute's centers in clusters so that the foundry can take advantage of visualization to determine what machines can be assigned to orders. To plot these centers, we map the input values into the range of -1 and $+1$ so that proper parallel coordinates can be used. Originating from the graphs, we also compare the same machines appearing in clusters that are produced by applying different clustering methods. These identical machines form the important basis to support the clustering results for the management of machines.
© 2006 Elsevier Ltd. All rights reserved.

Keywords: Clustering; IC packaging; Visualization

1. Introduction

Packaging is one of steps in the integrated circuits (ICs) manufacturing that can be broken into five steps in order of process: (1) starting substrate, (2) wafer fabrication, (3) wafer sort/test, (4) packaging, and (5) mark/final test. The first three steps are commonly referred to as front-end processing, while the last two are back-end. Upon completing the front-end processing, packaging is performed by processing the finished wafer whose surface contains many individual die, also called chips. General purposes of packaging include protecting ICs, making ICs easier to handle, and connecting ICs to the circuit outside. A packaging process starts with cutting wafers into individual chips by a wafer saw. Next, the chips are put onto the leadframe using a die bonder. Then, a wire bonder connects the electrical paths on the chip with the contact pads of the leadframe. After wire bonding, the chips are encapsulated using an injection molding process. Following the molding, the leads are tinned in a plating process, and the chips are marked. Finally, the leads are trimmed from the leadframe, formed into proper shape and the chip is cut out from the leadframe. The packaging process described above is shown in Fig. 1.

Continuing rising costs of wafer fabrication facility and labor have made popular the packaging foundries that benefit fabless IC design companies on the one hand, and wafer fabrication and foundry companies on the other. Despite the increasing popularity worldwide, an IC packaging foundry is exposed to the following challenges.

- Production scheduling is complicated because a wide variety of products are manufactured.
- Machine management is extremely challenging due to the particular fact that the *wire bond* machines not only outnumber other types of machines but outmatch the processing time in a packaging process.

[*] Corresponding author. Tel.: +886 4 23924505; fax: +886 4 23921742.
E-mail address: yanghh@ncit.edu.tw (H.-H. Yang).

0957-4174/$ - see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2006.05.009

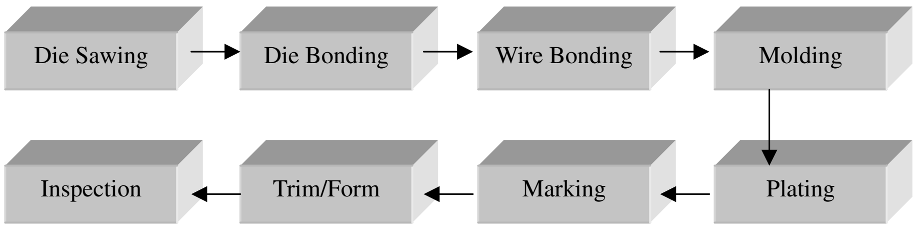

Fig. 1. An IC packaging process.

- Medium- and long-range forecast and shop floor control are difficult because of dynamic arrival of orders.
- Efforts to separate an incoming order into numerous smaller orders are required to deal with the situation where the order may contain a wide variety of chips to be packaged.
- Setup time of machines varies greatly; some take up to 16 h, while others need only 10–30 min.

Among the preceding challenges, management of the wire bond machines plays the most decisive role in improving a packaging foundry's productivity owing to their dominant quantities and processing time. In most cases, the foundry manages the machines simply by their serial numbers that are generally related to the year they were purchased, rarely by attributes to the management's advantage. However, assigning machines to incoming orders not only involves general temporal factors but also has to consider particular requirements such as whether orders are emergent or demand vast capacity, high quality. These requirements can not be cost-effectively met without clustering machines in a proper way.

Clustering is one of techniques that partition objects into clusters so that objects within a cluster have similar characteristics (or are close to each other), while objects in different clusters are most distinct from one another. The result of clustering depends heavily on the measure of the similarity between two objects. For some applications, however, it may be convenient or intuitive to measure the dissimilarity of objects instead of defining a similarity measure. The most common way to calculate the dissimilarity between two objects is to compute their *Euclidean distance*. A number of algorithms have been developed for clustering objects based on their similarity (or dissimilarity). The k-means algorithm (MacQueen, 1967) is a popular clustering technique that partitions a set of n objects into k clusters where cluster similarity is measured by the average value of the objects in the cluster. To use the k-means algorithm, the desired number of clusters k must be given as an input. We refer interested readers to Jain, Murty, and Flynn (1999) for a recent survey on clustering, and Grabmeier and Rudolph (2002); Jain and Dubes (1988) for techniques of cluster algorithms.

To represent relationships between data and then learn these relationships from the data, neural networks are a useful tool and often used to develop models that predict or classify an output as a response to a set of inputs to the trained network. One popular variant of neural networks is Kohonen's self-organizing map (SOM) (Kohonen, 1985; Kohonen, 1995) that projects high-dimensional input data onto a low-dimensional (usually one- or two-dimensional) array. This nonlinear projection produces a structured ordering of the input vectors that maps similar input objects to neighboring nodes in the two-dimensional map, which helps to understand the complex data structure or identify "clusters" of data in the high-dimensional space. Recall that the k-means algorithm, in spite of its easiness to implement and efficiency, requires an initial k that may be unavailable a priori and affect the final result. To remedy this drawback, researchers have developed some two-phase (or two-stage) methods (Balakrishnan, Cooper, Jacob, & Lewis, 1996; Jiang, Tseng, & Su, 2001; Kuo, Ho, & Hu, 2002; Yang & Liu, in press) that use available algorithms in the first phase to find k. The SOM algorithm has increasingly gained its popularity to be one of such algorithms used in the first phase.

Because of the importance of managing the wire bond machines, the objective of this paper is to apply clustering techniques to them so that a family of machines can be assigned to orders in the foundry's best interest. To cluster machines, we use the two-phase methodology proposed by Yang, Liu, and Su (submitted for publication) where SOM was used in the first phase to find k. Given this k, the k-means algorithm and minimum spanning tree (MST)-based clustering algorithm were used to find clusters in the second phase. We refer readers to Xu, Olman, and Xu (2001); Xu, Olman, and Xu (2002) for more details about using the MST-based clustering. Clustering naturally raises and needs to answer the questions of whether and how the results are good. Yang et al. (submitted for publication) compared and discussed the clustering results, but elaborated little in the context of managing machines related to the orders. Extending from the work of Yang et al. (submitted for publication), in this paper we focus more on management aspects than on comparing results. To understand what the clustering results reveal and may help, we use the parallel coordinate visualization (Berry & Linoff, 2004) that shows each attribute's centers in the clusters. Before plotting using parallel coordinate, input data are mapped into the range between +1 and −1. In view of clustering machines to improve management, Yang and Liu (in press) used a similar two-phase clustering methodology to observe how the clusters would be affected after removing outliers. However, Yang and Liu (in press) did not consider normalizing data whose real-life values may vary significantly, nor did they apply the MST-based clustering that enjoys advantages over the k-means algorithm with respect to irregular geometric shape of data. The rest of the paper is organized as follows. In Section 2, we briefly describe the k-means algorithm, SOM and MST. In Section 3, we describe the data and discuss how the visualization may help the management of machines given clustering results. We present conclusions in Section 4.

2. Background

In this section, we will introduce basics of the k-means algorithm, SOM and MST. We begin with the k-means algorithm.

2.1. k-means algorithm

The k-means algorithm finds a set of k clusters so that the total distance, usually Euclidean, is minimized between the input objects x_i and its closest cluster c_j. The outline of the k-means algorithm is given as follows:

1. Choose k centroids arbitrarily for each cluster c_j, $j \in [1,k]$.
2. Assign each object to the cluster whose centroid is closest to the object.
3. Compute the centroid of each cluster c_j, $j \in [1,k]$.
4. Repeat Steps 2 and 3 until no objects change between clusters.

Major attractiveness of the k-means algorithm is its computational efficiency that requires polynomial time of $O(t\,k\,n)$, where t is the number of iterations, k is the number of clusters, and n is the number of objects. Because n dominates t and k in most practical applications, the k-means algorithm requires approximate $O(n)$ time. However, drawbacks of the k-means algorithm include: (1) it often converges to a local optimum, (2) it favors clusters with convex shapes, and (3) it is sensitive to the presence of outliers and initial solution.

2.2. SOM algorithm

A typical SOM network contains two layers, i.e., the input layer and output (Kohonen) layer, where the input layer is fully connected to a two-dimensional Kohonen layer (Fig. 2). Assume there are N input vectors, each vector contains K attributes, and the size of the Kohonen layer is M that is usually a square matrix. As shown in Fig. 2, input layer contains the set of input vector $x_i = (x_{i1}, x_{i2}, \ldots, x_{iK})$, $i \in [1,N]$, and corresponding to this x_i in the output layer is the weight vector $w_m = (w_{m1}, w_{m2}, \ldots, w_{mK})$, $m \in [1,M]$. During the training process, each input vector is presented to the network through the processing nodes in the input layer. As the training process proceeds, the values of the weight vector are adjusted according to the topological relations in the input vector. The node with the minimum weight vector is the *winner* and weight values of this winner's neighboring nodes are adjusted to be closer to the value of the input vector. Euclidean distance is the most popular way to measure the distance between input vectors and weight vectors.

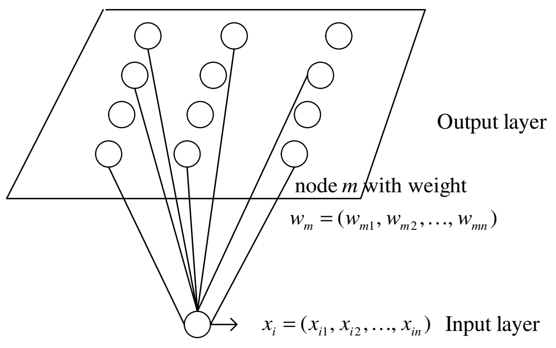
Output layer
node m with weight
$w_m = (w_{m1}, w_{m2}, \ldots, w_{mn})$
$x_i = (x_{i1}, x_{i2}, \ldots, x_{in})$ Input layer

Fig. 2. Kohonen's SOM network.

Let discrete time t (epoch) be an index such that $x_i(t)$, $t = 0, 1, \ldots$, is presented to network at time t, and $w_m(t)$ is the weight vector computed at time t. The SOM algorithm works as follows:

1. Present input vector $x_i(t)$ randomly.
2. Compute the Euclidean distance for each node in the output layer.

   $$\|x_i(t) - w_m(t)\|, \quad m \in [1,M].$$

3. Find the winner with the minimum distance.

   $$\|x_i(t) - w_c(t)\| = \min \|x_i(t) - w_m(t)\|, \quad m \in [1,M].$$

4. Update weight vectors of the neighboring nodes of the winner.

   $$w_m(t+1) = w_m(t) + \alpha(t)h(t)[x_i(t) - w_m(t)].$$

In Step 4, $\alpha(t)$ is the learning rate that controls the overall magnitude of the correction to the weight vectors and is reduced monotonically during the training process; $h(t)$ is the neighborhood function that controls the extent to which $w_m(t)$ is allowed to adjust in response to an input most closely resembling $w_w(t)$.

2.3. Minimum spanning tree

Given a graph G, a spanning tree T of this graph G is a connected acyclic subgraph that connects all nodes. A minimum spanning tree is the spanning tree whose total edge length is the smallest. Zahn (1971) proposed a clustering algorithm that initially constructs the MST of the given data, and then deletes $k-1$ edges in ascending order of edge lengths to generate k trees, each of which corresponds to a cluster. The intuition behind this method is that the longest edges separate clusters, while the shortest edges connect close data points within clusters.

3. The data and results

We collect 366 records of data from a Taiwanese foundry, and choose six attributes as follows: (1) Prod_Qty, (2) Prod_Time, (3) Small_Stops, (4) Standby, (5) Changeover_Time, and (6) Breakdowns. These attributes are chosen after consulting with the foundry's staff. For brevity, we omit the detailed explanations. For later use, we provide each attribute's average of 366 records as follows: Prod_Qty = 20439.89, Prod_Time = 648.92, Small_Stops = 280.54, Standby = 123.21, Changeover_Time = 30.12, Breakdowns = 46.43. As shown in Table 1, the absolute value of Prod_Qty differs largely from that of Standby or Breakdowns. Therefore, we use two popular methods to normalize data, namely, min–max normalization (min–max for short) and z-score normalization (z-score for short). Given a vector $V = (v_1, v_2, \ldots, v_n)$, the min–max normalization of v_j is as follows:

$$\text{min}-\text{max}(v_j) = \frac{v_j - \text{min}}{\text{max} - \text{min}},$$

where max (or min) denotes the maximum (or minimum) of V.

Similarly, the z-score normalization is

$$z\text{-score}(v_j) = \frac{v_j - \bar{V}}{\sigma_V},$$

where $\bar{V}$ is the mean of V, and σ_V is the standard deviation of V.

With data normalization in place, we begin our two-phase methodology. According to Yang et al. (submitted for publication), the result of using the SOM to cluster data is shown in Fig. 3, where the size of Kohonen layer is 19×19, neighborhood function h is eight, learning rate α is 0.1, and learning epoch is 70. From Fig. 3, it is determined that the number of cluster is seven. Given this number, we can proceed to the second phase.

Because we use two other types of data normalizations, there are six combinations of clustering results. Table 2 shows the number of records in each cluster of each combination. Apparently, as pointed out by Yang et al. (submitted for publication), the k-means algorithm produces more even clusters after data normalization. To take advantage of visualization, we map the input values of attributes into the range between -1 and $+1$ and use parallel coordinates to plot centers of seven clusters identified. The center of a cluster is the most average member of this cluster. Among the six combinations, take Table 3 that uses k-means on raw data for example. Given Table 3, we use Table 4 to show the mapped values and then plot centers of clusters in Fig. 4. In addition to Fig. 4, we present the centers of clusters of the other two combinations in Figs. 5 and 6. Note that we exclude the figures produced by the MST because the extremely imbalanced distributions of records in clusters shed little light on assigning machines to orders.

One might question that these centers can simply be shown in tables rather than in figures; however, taking

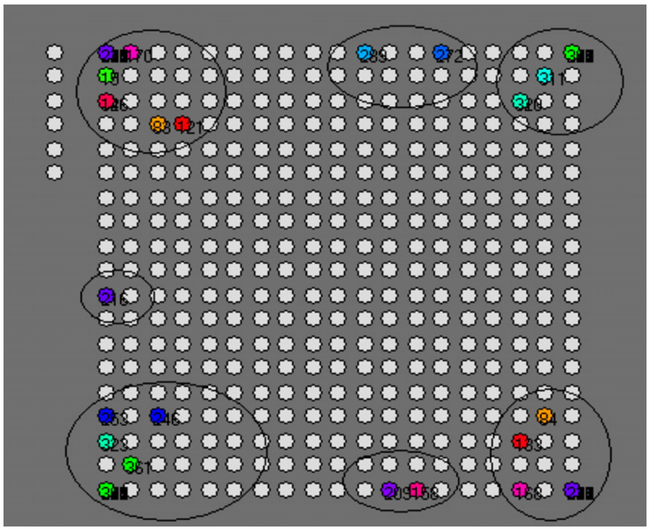
Fig. 3. The clustering result of raw data by SOM.

Table 2
Number of records in each cluster by MST and k-means

Cluster	Method					
	Raw data		z-score normalization		min–max normalization	
	MST	k-means	MST	k-means	MST	k-means
1	10	31	360	11	360	7
2	1	1	1	147	1	26
3	1	110	1	11	1	35
4	1	11	1	54	1	79
5	1	61	1	1	1	138
6	3	151	1	88	1	80
7	349	1	1	54	1	1

Table 1
Examples of raw data

Machine	Attribute					
	Prod_Qty	Prod_Time	Small_Stops	Standby	Changeover_Time	Breakdowns
5351WB	38522	450.22	85.15	11.8	2.87	5.34
5352WB	38467	634.78	88.61	24.01	2.87	10.69
5354WB	32356	398.98	31.69	17.82	0	9.53
5353WB	40855	432.61	35.14	7.75	0	1.96
5355WB	32453	409.7	31.49	18.22	0	6.69
.	.	.	.	.	.	.

Table 3
Attributes' average values in clusters by using k-means on raw data

Attribute	Cluster						
	1($n = 31$)	2($n = 1$)	3($n = 110$)	4($n = 11$)	5($n = 61$)	6(n =151)	7($n = 1$)
Prod_Qty	8857.645	49 728	15 452.88	37 003.55	28 426.93	21 556.08	60 819
Prod_Time	310.0497	801.09	552.5474	981.8627	815.8456	690.7896	1437.32
Small_Stops	473.5552	153.75	296.9441	189.6182	218.4433	261.5744	272.58
Standby	109.2758	112.24	130.9696	92.04182	112.6313	126.8417	151.85
Changeover_Time	33.84	23.55	33.96873	28.55545	25.39	28.62921	29.38
Breakdowns	50.77968	37.33	46.63536	26.55091	42.72361	48.27616	64.75

Table 4
Attributes' mapped values in clusters by using k-means on raw data

Attribute	Cluster						
	1($n = 31$)	2($n = 1$)	3($n = 110$)	4($n = 11$)	5($n = 61$)	6($n = 151$)	7($n = 1$)
Prod_Qty	0.114572	0.811008	0.226955	0.594182	0.448035	0.330955	1
Prod_Time	0.182835	0.538793	0.358623	0.669836	0.54949	0.458836	1
Small_Stops	0.293295	0.079349	0.175144	0.103344	0.122628	0.151482	0.158845
Standby	0 .470092	0.483817	0.570541	0.390294	0.485629	0.551427	0.667222
Changeover_Time	0.364223	0.253471	0.365609	0.307345	0.273275	0.308139	0.31622
Breakdowns	0.124321	0.090071	0.113768	0.062622	0.103806	0.117946	0.159897

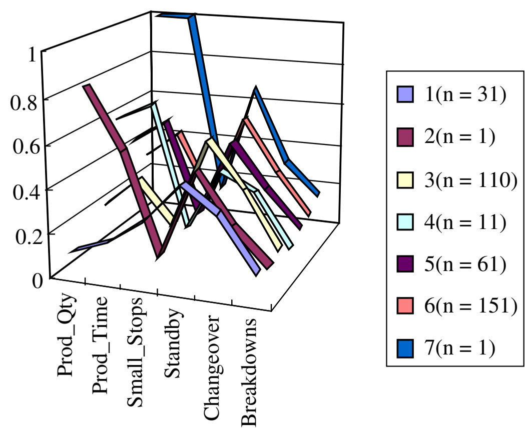

Fig. 4. Centers of seven clusters by using k-means on raw data. (For interpretation of the references to colour in this figure, the reader is referred to the web version of this article.)

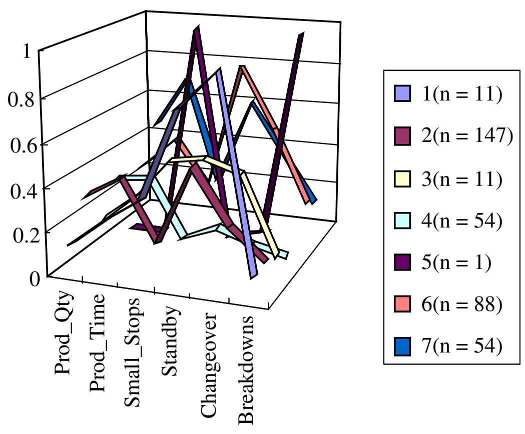

Fig. 5. Centers of seven clusters by using k-means on data with z-score normalization. (For interpretation of the references to colour in this figure, the reader is referred to the web version of this article.)

advantage of visual effects enables managers to make the observations related to orders promptly and easily. First, consider the attribute Prod_Qty that means the production quantity of the machine. Comparing Figs. 4–6, we find that the 11 machines in cluster 4 (in paleturquoise) in Fig. 4 seem to form a good family for orders that demand high outputs. In this case, the average Prod_Qty of the 11 machines is 37 003.55, which is nearly two times of that of the total average, 20 439.89. Their high average production quantities justify their candidacy for orders of high outputs. In Fig. 4, one might also notice that in terms of Prod_Qty, cluster 4 is smaller than clusters 2 (in maroon) and 7 (in blue) where both contain only one record. The implication is that these two machines may also be considered but it is at the discretion of shop floor supervisors who better understand whether the machines constitute "outliers." Second, for emergent time-based orders, the 54 machines in cluster 4 in Fig. 5 (or the 79 machines in cluster 4 in Fig. 6) can be considered. From Fig. 5 we observe Small_Stops, Standby, Changeover_Time and Breakdowns that largely affect an order's completion time are relatively low. In this case, the average Small_Stops is 100.34; Standby is 32.06; Changeover_Time is 5.09; and Breakdowns is 9.05. Comparing to the averages of total records, which are 280.54, 123.21, 30.12, 46.43, we suggest that the machines in this cluster be more appropriate for emergent orders. What deserves to be further investigated is whether the low Prod_Qty (19486.02) can be increased so that high output in short time can be simultaneously achieved.

We have remarked above that cluster 4, not only in Fig. 5 but also in Fig. 6, is suitable for emergent orders. After examining the raw data, we find that the 54 machines

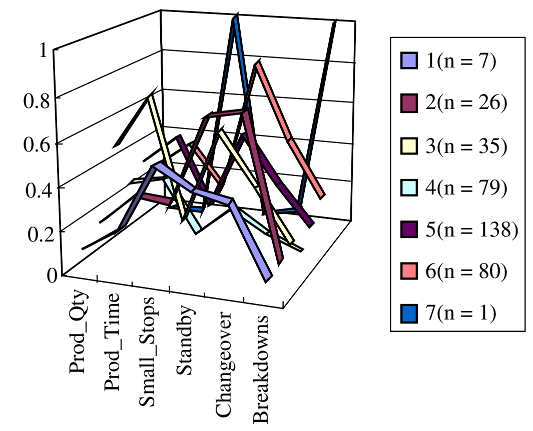

Fig. 6. Centers of seven clusters by using *k*-means on data with min–max normalization. (For interpretation of the references to colour in this figure, the reader is referred to the web version of this article.)

Table 5
List of numbers of identical machines in clusters

Cluster	Method		
	k-means with *z*-score normalization	*k*-means with min–max normalization	Number of identical machines
1	11	26(2)	10
2	147	138(5)	109
3	11	7(1)	6
4	54	79	54
5	1	1(7)	1
6	88	80	67
7	54	35(3)	35

in both clusters actually are identical. This leads us to investigate an interesting question of whether some machines tend to be grouped together even different clustering algorithms are used. Table 5 shows the numbers of the same machines appearing in the clusters by using two types of data normalizations, where the numbers in parentheses represent the cluster number using this type of normalization. According to Table 5 we find that the degree of duplicate machine is rather high. This high duplicate should reinforce the foundry's confidence in relying on the clustering results to assign the machines to various forms of orders. Moreover, we find that some serial numbers of machines in the same cluster follow close sequence. Because the serial number of a machine is generally assigned according to the year purchased, close sequence of some machines frequently appearing in the same cluster may reveal the information such as whether the year to purchase machines affects the performance.

4. Conclusions

In this paper, we apply a two-phase methodology to cluster 366 records of the wire bond machines for a Taiwanese IC packaging foundry, where six attributes of each machine are chosen to cluster. The purpose of clustering in this paper is to help the foundry assign machines more effectively to orders that arrive in various types ranging from time-based to quality-demanding and highly dynamically in nature. Given the clusters obtained and taking advantage of a visualization, we plot centers of clusters using parallel coordinates that provide an easier outlook for managers to assign. Because values of some attributes are as large as thousands while some are only in tens, we need to map these values into the range between -1 and $+1$ so that proper parallel coordinates can be used to plot. Despite the visual advantage founded on mapping values, we need to point out that the drawback remains because given the graphs we still need to find the original values to know more about the details. However, this drawback is less significant compared to the benefits resulting from more effective management of machines. The last result of the paper is that we present the numbers of identical machines appearing in the same clusters when different clustering methods are used. The implication is that the clusters obtained are more robust as more identical machines are present.

References

Balakrishnan, P. V., Cooper, M. C., Jacob, V. S., & Lewis, P. A. (1996). Comparative performance of the FSCL neural net and *k*-means algorithm for market segmentation. *European Journal of Operational Research, 93*(3), 346–357.

Berry, M. J., & Linoff, G. S. (2004). *Data Mining Techniques*. Indianapolis, Indiana: Wiley Publishing, Inc.

Grabmeier, J., & Rudolph, A. (2002). Techniques of cluster algorithms in data mining. *Data Mining and Knowledge Discovery, 6*, 303–360.

Jain, A. K., & Dubes, R. C. (1988). *Algorithms for clustering data*. Upper Saddle River, NJ: Prentice Hall.

Jain, A. K., Murty, M. N., & Flynn, P. J. (1999). Data clustering: A review. *ACM Computer Survey, 31*(3), 264–323.

Jiang, M. F., Tseng, S. S., & Su, C. M. (2001). Two-phase clustering process for outliers detection. *Pattern Recognition Letters, 22*(6–7), 691–700.

Kohonen, T. (1985). The self-organization map. *Proceedings of IEEE, 73*, 1551–1558.

Kohonen, T. (1995). *Self-organizing maps*. Springer-Verlag.

Kuo, R. J., Ho, L. M., & Hu, C. M. (2002). Integration of self-organizing feature map and *k*-means algorithm for market segmentation. *Computers and Operations Research, 29*, 1475–1493.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*, pp. 281–297.

Xu, Y., Olman, V., & Xu, D. (2001). Minimum spanning tree for gene expression data clustering. *Genome Informatics, 12*, 24–33.

Xu, Y., Olman, V., & Xu, D. (2002). Clustering gene expression data using a graph-theoretic approach: An application of minimum spanning tree. *Bioinformatics, 18*(4), 536–545.

Yang, H.-H., & Liu, T.-Z. (in press) Clustering with removing outliers – A case of an IC packaging company. International Journal of Advanced Manufacturing Technology (DOI 10.1007/s00170-005-0137-3).

Yang, H.-H., Liu, T.-Z., & Su, H.-T. (submitted for publication). Two-phase clustering – a case of a Taiwanese IC packaging foundry. Pattern Recognition Letters.

Zahn, C. T. (1971). Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Transactions on Computers, C-20*, 68–86.