A Fast Algorithm for K-Means Optimization using Pillar Algorithm

Ali Ridho Barakbah and Yasushi Kiyoki, *Member, IEEE*

Abstract— **The ability of K-means to cluster a kind of huge data very quickly often pays to the incorrect clustering results because of its designated initial centroids as cluster centers which are generated randomly. However, the efforts to improve the precision of K-means clustering results may take a highly execution time by optimizing the determination of initial centroids for K-means. This paper presents a fast algorithm for K-means optimization by improving our Pillar algorithm. The Pillar algorithm [19] is inspired by the thought process of determining a set of pillars' locations in order to make a stable house or building. The algorithm considers the pillars' placement which should be located as far as possible from each other to withstand against the pressure distribution of a roof, as identical to the number of centroids amongst the data distribution. Hence, the Pillar algorithm designates positions of initial centroids by using the farthest accumulated distance between them. This algorithm is very effective to position the initial centroids for K-means and improve the precision of the clustering results. However, the algorithm takes highly computational time for clustering huge data which often have many outliers, since its complexity O((*k*+*h*+1) n) (where *k*=number of clusters, *h*=number of outliers, and *n*=number of data items) to position the initial centroids. In this paper, we reduce the complexity of our previous work Pillar algorithm by excluding the designated initial centroids' neighbors from next iterations so that the complexity will decrease in line with iterations and speed up the execution time. The performance of our improved Pillar algorithm is examined in the precision rate and computational time with several benchmark datasets as well as image data and compared the existing algorithms. The experimental results show the improved solution using the proposed approach.**

I. Introduction

CLUSTERING is a widely used knowledge discovery technique to classify unsupervised objects in the same groups by considering their similarities. A good cluster is constructed when the members of the cluster have a high degree of similarity of each other (internal homogeneity) and are not like members of other clusters (external homogeneity) [4, 9]. It means that the process to define a mapping $f{:}D \rightarrow C$ from some data $D=\{d_1,d_2,..,d_n\}$ to some clusters $C=\{c_1,c_2,..,c_n\}$ on similarity between d_i [19]. The applications of clustering are diversely in many fields such as data mining, pattern recognition, image classification, biological sciences, marketing, city-planning, document retrievals, etc.

The most well known, widely used and fast methods for clustering is K-means algorithm developed by Mac Queen in 1967. The simplicity of K-means made this algorithm used in various fields. K-means algorithm is a partitioning clustering method that separates data into *k* mutually excessive groups. Through such the iterative partitioning, K-means clustering minimizes the sum of distance from each data to its clusters. K-means clustering is very popular because of its ability to cluster a kind of huge data, and also outliers, quickly and efficiently. It remains a basic framework for developing numerical or conceptual clustering systems because various possibilities of distance and prototype choice [2].

However, K-means clustering is very sensitive to the designated initial starting points as cluster centers. K-means clustering generates initial clusters randomly. If a randomly designated initial starting point closes to a final cluster center, then K-means clustering can find the final cluster center. It however is not always. If a designated initial point is far from the final cluster center, it will lead to incorrect clustering results [11]. Because of initial starting points generated randomly, K-means clustering does not guarantee the unique clustering results [13]. K-means clustering is difficult to reach global optimum, but only to one of local minima [7]. The better results of K-means clustering can be achieved after computing more than one times. However, it is difficult to decide the execution limit, which gives the best performance [17]. The cluster initialization algorithms for K-means tried to apply heuristic mechanisms to avoid the uncertainty of *n* times trial of K-means execution.

Several methods proposed to solve the cluster initialization for K-means clustering. A recursive method for initializing the means by running *k* clustering problems is discussed by Duda and Hart (1973). A variation of this method consists of taking the entire data into account and

This work was supported in part by the Japanese Ministry of Education, Culture, Sports, Science and Technology, and Keio University, under Grant Monbukagakusho scholarship.

A.R. Barakbah is a lecturer and head of Soft Computing Laboratory in Information Technology Department, Electronic Engineering Polytechnic Institute of Technology, Surabaya, Indonesia. Currently, he is a doctoral student at Graduate School of Media and Governance, Keio University, Japan, and belonging to Multi Database and Multimedia Database Laboratory, Keio University, Shonan Fujisawa Campus, Japan (corresponding author to provide phone: +81-80-3516-4979; fax: +81-466-486-093; e-mail: ridho@mdbl.sfc.keio.ac.jp).

Y. Kiyoki is a professor of the Faculty of Environmental Information, Keio University, Japan. He is a supervisor of Multi Database and Multimedia Database Laboratory, Keio University, Shonan Fujisawa Campus, Japan (e-mail: kiyoki@sfc.keio.ac.jp).

then randomly perturbing it *k* times [13]. Bradley and Fayyad (1998) proposed an algorithm that refines initial points by analyzing distribution of the data and probability of data density [3]. Penã et al. (1999) presented empirical comparison for four initialization methods for K-means clustering those are random, Forgy approach, Mac Queen approach, and Kaufman approach [5].

Our previous works have been dealing for K-means Optimization. Barakbah and Helen (2005) presented a new algorithm, called as Optimized K-means, that spreads the initial centroids in the feature space so that the distances among them are as far as possible [14]. Barakbah et al. (2005) presented the optimization of initial starting points for K-means using Simulated Annealing [15]. Barakbah (2006) proposed a new algorithm to optimize the initial centroids for K-means by separately locating them as far as possible in data distribution [16]. By considering the computation of K-means optimization, Barakbah and Arai presented a fast algorithm for designating the initial centroids by embedding hierarchical algorithm into K-means clustering [17].

In this paper we propose a fast algorithm for optimizing the initial centroids for K-means by improving our previous work Pillar algorithm presented in [19]. The Pillar algorithm [19] is inspired by the designation of pillars' placement in a house or building. The algorithm considers the pillars which should be located as far as possible from each other to withstand against the pressure distribution of a roof, as the number of centroids amongst the data distribution. Therefore, the Pillar algorithm determines the position of initial centroids by calculating the accumulated distance metric between each data and all previous centroids, and then, a data point which has the maximum distance will be selected. This algorithm is also robust whereby it has a mechanism to avoid outlier data being chosen as the initial centroid. However, the algorithm takes highly computational time for clustering huge data which often have many outliers. In this paper, we improve our Pillar algorithm by modifying the status of each initial centroids' neighbors after applying the ourlier detection mechanism. The improvement is very essential to speed up the computional time of K-means optimization by Pillar algorithm, especially for huge and complex data sets.

We organize this paper as follows. In Section 2, the Pillar algorithm is described. Section 3 discusses the improvement of our Pillar algorithm. Section 4 describes the validity measurements those are used in the experiments. Section 5 describes the experimental results using several benchmark datasets as well as image data with several comparing algorithms, and then followed by concluding remarks in Section 6.

II. Pillar Algorithm

This section describes the basic concepts of our Pillar algorithm to designate the initial centroids, a mechanism of outlier detection, and the improvement of the Pillar algorithm. This section also contains a complexity analysis of the proposed algorithm and a sequential logical flow for the proposed approach.

A. Basic Concept

The K-means algorithm generates the initial centroids randomly and fails to consider a spread out placement of them spreading within the feature space. In this case, the initial centroids may be placed so close together that some become inconsequential. Because of this, the initial centroids generated by K-means may be trapped in the local optima. The Pillar algorithm proposed a method of placing the initial centroids whereby each of them has a farthest accumulated distance between them.

The Pillar algorithm is inspired by the thought process of determining a set of pillars' locations in order to make a stable house or building. Fig. 1 illustrates the locating of two, three, and four pillars, in order to withstand the pressure distributions of several different roof structures composed of discrete points. It is inspiring that by distributing the pillars as far as possible from each other within the pressure distribution of a roof, the pillars can withstand the roof's pressure and stabilize a house or building.

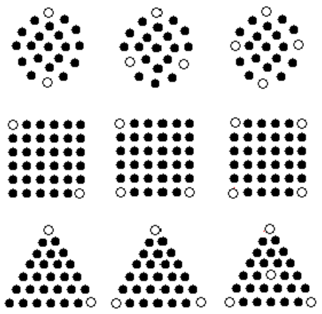
Fig. 1. Illustration of locating a set of pillars (white points) withstanding against different pressure distribution of roofs.

The algorithm considers the pillars which should be located as far as possible from each other to withstand against the pressure distribution of a roof, as number of centroids among the gravity weight of data distribution in the vector space. Therefore, the Pillar algorithm designates positions of initial centroids in the farthest accumulated distance between them in the data distribution.

B. Determining Initial Centroids

First of all, the grand mean of data points is calculated as the gravity center of the data distribution. The distance

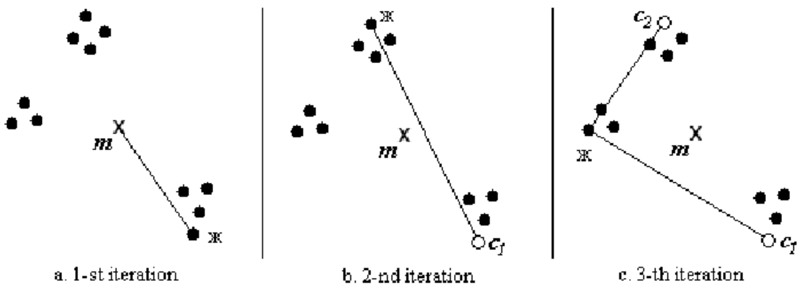

Fig. 2. Selection for several candidates of the initial centroids.

metric D (let D^1 be D in this early step), is then created between each data point and the grand mean. A data point which has the highest distance in D^1 will be selected as the first candidate of the initial centroid ж. Fig. 2a illustrates m as the grand mean of data points and ж which is has the farthest distance to m is the candidate of the first initial centroid.

If ж is not an outlier, it will be promoted to the first initial centroid c_1. We then recalculate D (D^2 in this step), which is the distance metric between each data points and c_1. Starting from this step, we use the accumulated distance metric DM and assign D^2 to DM. The construction of DM is started from D^1.

To select a candidate for the second initial centroid, the same mechanism is applied using DM instead of D. The data point with the highest distance of DM will be selected as the second initial centroid candidate ж, as shown in Fig. 2b. If ж is not classified as an outlier, it becomes c_2. To select a next ж for the candidate of the rest initial centroids, D^t (where t is the current iteration step) is recalculated between each data points and c_{t-1}. The D^t is then added to the accumulated distance metric DM ($DM \leftarrow DM + D^t$). This accumulation scheme can avoid the nearest data points to c_{t-1} being chosen as the candidate of the next initial centroid. It consequently can spread out the next initial centroids far away from the previous ones. The data points with the highest distance in DM will then be selected as ж, as shown in Fig. 2c.

If ж is not an outlier, it will become c_t. The iterative process guarantees that all initial centroids are designated.

C. Outlier Detection Mechanism

To be selected as the initial centroids, the data point candidate must not be categorized as outlier. We identify an outlier by considering the number of neighbor points within the neighborhood boundary. Let n be the number of data points and k be the number of clusters, we set the number of neighbors *nmin* by using a probabilistic parameter α to the average members of clusters n/k. For assuming the neighborhood boundary *nbdis*, we apply a threshold β to the highest distance in D^1, as shown in Fig. 4.

To get several data points to be the neighbors inside the boundary, it needs to calculate a distance metric between all data points and ж. However, it can utilize distance metric D for the next iteration step. In case shown in Fig. 4, the distance metric between all data points and ж can be acquired from D^2. If the number of neighbors inside the boundary is lower than *nmin*, ж will be classified as an outlier. When ж is considered as an outlier, it will be redetermined by selecting the second highest distance in D^1. This outlier detection mechanism is iteratively executed until ж can be considered as the initial centroid.

III. Improvement of Pillar Algorithm

The Pillar algorithm is very effective to position the initial centroids for K-means and improve the precision of the clustering results. However, the algorithm takes highly computational time for clustering huge data which often have many outliers, since its complexity O((k+h+1) n) (where k=number of clusters, h=number of outliers, and n=number of data items) to position the initial centroids. In this paper, we reduce the complexity of our previous work

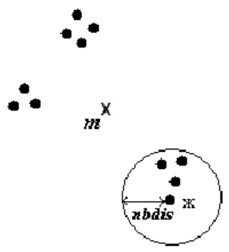

Fig. 4. An illustration of the neighborhood boundary.

Pillar algorithm by excluding the designated initial centroids' neighbors from next iterations so that the complexity will decrease in line with iterations.

A. Excluding initial centroids' neighbors

In the Pillar algorithm, when a data item is chosen as a initial centroid candidate ж, the ourlier detection mechanism is applied by identifying the neighbors. If the number of neighbors inside the boundary in *nbdis*, as shown in Fig. 4, is same or higher than *nmin*, ж will be promoted to an initial centroid c_t in iteration t. Unlike our previous Pillar algorithm in which D^{t+1} involves distance calculation between all data items to c_t in order to designate next initial centroid c_{t+1}, we improve the algorithm by reducing number of distance calculation in D^{t+1}. When ж is promoted to be c_t, the neighbors of c_t inside *nbdis* is noted. These neighbors are supposed to belong to c_t and do not need to involve in the distance calculation D^{t+1}. By excluding them in D^{t+1}, number of distance calculations can be reduced for the next steps. It will decrease the complexity and speed up the execution time for designating all initial centroids.

B. Algorithm Complexity

The time complexity of the Pillar algorithm is $O((k+h+1)\,n)$, where k is number of clusters, n is number of data items, and h is number of outliers in the data set. In a case in which there is no outliers in the data set, the complexity becomes $O((k+1)\,n)$, or equal to $O((t+1)\,n)$, where t is the number of iterations, since selecting an initial centroid for each k takes one iteration. For worse case in which there are number of outliers close to n ($h \approx n$), the complexity becomes $\approx O(n^2)$.

With the improvement of Pillar algorithm by excluding the initial centroids' neighbors for each designated initial centroids, number of data n items will decrease in line with the iterations. When these initial centroids' neighbors are excluded and not involved in the distance calculation for next steps, n in the iterations will be decreased. The complexity of improved Pillar algorithm is $O(n+(h_1.n_1)+\ldots+(h_k.n_k))$ where $n_k<\ldots<n_1<n$, n_i is the rest of number of data items after excluding the i-th designated initial centroids' neighbors and h_i is number of ourliers before the i-th designated initial centroids. By this improvement, in a case in which there is no outliers, the complexity becomes $O(n+n_1+\ldots+n_k)$. The complexity will be $\approx O(n^2)$ for worse case in which number of outliers close to n.

IV. Validity Measurements

For performance analysis, we use several validity measures for clustering results; those are variance within cluster, variance between clusters, sum of squared error, standard deviation validity index, centroid proximity index, error ratio and execution time.

A. Variance Analysis

Variance constraint [9] can express the density of the clusters with the variance within cluster and the variance between clusters [6, 13]. The ideal cluster has minimum variance within cluster to express internal homogeneity and maximum variance between clusters to express external homogeneity [12]. Let $X=\{x_i \mid i=1,\ldots,N\}$ be data set, $S=\{s_i \mid i=1,\ldots,k\}$ be clustered X where $M \in X$ is $M_i=\{m_{ij} \mid j=1,\ldots,n(s_i)\}$ as members of s_i, variance within cluster can be defined as follows:

$$v_w = \frac{1}{N-k}\sum_{i=1}^{k}\left(n(s_i)-1\right)v_i^2 \tag{3}$$

where N is number of data points, k is number of clusters, and n_i is number of members in i-th cluster, while v_i is given as:

$$v_i^2 = \frac{1}{n(s_i)-1}\sum_{j=1}^{n(s_i)}\left(\underset{\in s_i}{m_{ij}} - \bar{s}_i\right)^2 \tag{4}$$

where m_j is members of i-th clusters.

Variance between clusters then can be defined as follows:

$$v_b = \frac{1}{k-1}\sum_{i=1}^{k} n(s_i)\left(\bar{s}_i - \bar{x}\right) \tag{5}$$

B. Sum of Squared Error

The most widely used criterion to quantify cluster homogeneity is the sum of squared error criterion [9]. It can be defined as:

$$SSE = \frac{\sum_{i=1}^{k}\sum_{j=1}^{n(s_i)}\left\|m_{ij}-\bar{s}_i\right\|^2}{N} \tag{6}$$

C. Standard Deviation Validity Index

The standard deviation validity index is defined based on the concepts of the average scattering for clusters and total separation between clusters [8]. It consists of average scattering for clusters and total separation between clusters. The average scattering for clusters is defined as:

$$scat(k) = \frac{1}{k}\sum_{i=1}^{k}\frac{\left\|\sigma(s_i)\right\|}{\left\|\sigma(x)\right\|} \tag{7}$$

The definition of total scattering (separation) between clusters is given as follows:

$$dis(k) = \frac{D_{\max}}{D_{\min}}\sum_{i=1}^{k}\left(\sum_{j=1, j\neq i}^{k}\left\|s_i - s_j\right\|\right)^{-1} \tag{8}$$

where D_{max} = max($\|s_i$-$s_j\|$) is the maximum distance between centroids, and D_{min} = min($\|s_i$-$s_j\|$) is the minimum distance between centroids. The standard deviation validity index for k clusters can be defined as:

$$SDVI(k) = a \, . \, scat(k) + dis(k) \tag{9}$$

where a is a weighting factor equal to $dis(c_{max})$ where c_{max} is the maximum number of input clusters.

D. Centroid Proximity Index

We propose this kind of validity measurement, called centroid proximity index, to analyze the closeness of the final centroids of the clustering result to the centroids of the real data sets. The CPI is defined as:

$$CPI = \min \sum_{i=1}^{k} (\|c_i - r_i\|) \tag{10}$$

where c_i is i-th final centroid of clustering result and r_i is i-th real centroid of datasets.

E. Error ratio

Because we use several benchmark data sets which are already classified, the error ratio expresses the error rate of clustering results compared to the classified labels of the datasets. The error ratio is defined as:

$$Error = \frac{numberofincorrectresults}{numberofdataitems} \tag{11}$$

V. Experimental Results

To establish practical applicability of our Pillar algorithm with its improvement, we made a series of experiments and tested its performance on several benchmark datasets, mostly from UCI repository; those are Ruspini[1], Fossil [1], Iris, New Thyroid, Wine, Glass, Heart, and Ionosphere [18] datasets. For comparison purpose, we use the plain data of each data set without applying the normalization.

We conducted the performance comparison between the improved Pillar algorithm and several approaches of initial centroids optimization for K-means; those are Forgy approach, Mac Queen approach, Kaufman approach, Refinement approach [3], MDC [16] and K-means using random initialization. For Forgy, Mac Queen and Refinement approach, because those approaches cannot give the unique clustering results, we made 10 times experiments and noticed their average results. We set α=0.1. We remove parameter β and directly set *nbdis* with standard deviation of the data items to their grand mean for representing the density of data distribution. In addition we also conducted the comparison between our algorithm in this paper and several clustering algorithms those are hierarchical clustering (Single, Linkage, Linkage, and Average Linkage), and possibilistic clustering (Fuzzy C-Means). We run all programs under Matlab 8.0, with computer specifications: Intel Core2 Duo, 2 GHz MHz, 3 GB RAM. We use built-in Matlab toolbox for Hierarchical algorithms. For Fuzzy C-Means, we used a program developed by David Corney (2000). Aside from that, we developed our own programs for the rest of algorithms.

[1] built-in data object in both R and S-plus statistics packages.

Table 1-Table 8 shows the comparison results of validity measurements for each data set. From those tables, the precision of our improved Pillar algorithm can reach best performance in 3 validity measurements on average and outperform the other comparing algorithms. Fig. 5 and Fig. 6 show the total best achievement of each clustering algorithm in all validity measurements. In those figures, our improved Pillar algorithm shows better performance to the comparing algorithms.

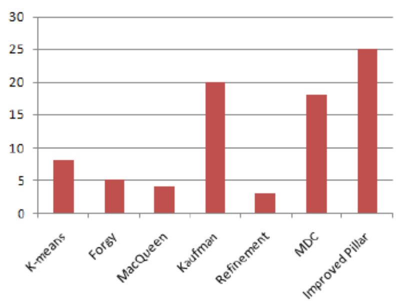

Fig. 5. Comparison of the total best achievement of each Kmeans optimization algorithm in all validity measurements

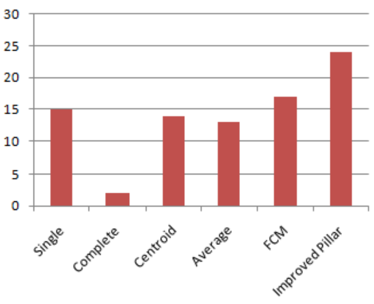

Fig. 6. Comparison of the total best achievement of each clustering algorithm in all validity measurements

Table 9 shows the comparison of computational time for the K-means optimization algorithms. Our improved Pillar algorithm is able to speed up the computational time of Kmeans optimization. The improved Pillar algorithm outperformed the other comparing algorithm in all datasets.

We also made experiments of clustering for image data using SIMPLIcity dataset of Wang et al. [20]. Fig. 7 shows one of visual comparisons of image clustering result using

TABLE I
VALIDITY MEASUREMENT IN RUSPINI DATA SET

Algorithm	v_w	v_b	*SSE*	*SDVI*	*CPI*	*Error(%)*
K-means with random init.	284.50	78788.05	269.32	0.52	23.69	12.8
K-means with Forgy init.	388.01	77183.15	367.31	0.71	51.20	9.33
K-means with Mac Queen init.	436.15	77704.16	412.89	0.70	59.00	16.27
K-means with Kaufman init.	181.42	78190.89	171.75	0.29	0.00	0
K-means with Refinement init.	333.12	76886.69	315.35	0.46	34.28	2.53
K-means with MDC init.	181.42	78190.89	171.75	0.29	0.00	0
Single Linkage	181.42	78190.89	171.75	0.29	0.00	0
Complete Linkage	229.01	76572.48	216.79	0.32	9.47	4
Centroid Linkage	181.42	78190.89	171.75	0.29	0.00	0
Average Linkage	181.42	78190.89	171.75	0.29	0.00	0
Fuzzy C-Means	181.42	78190.89	171.75	0.29	0.00	0
K-means with imprv Pillar algorithm	181.42	78190.89	171.75	0.29	0.00	0

TABLE II
VALIDITY MEASUREMENT IN FOSSIL DATA SET

Algorithm	v_w	v_b	*SSE*	*SDVI*	*CPI*	*Error(%)*
K-means with random init.	42.92	6696.84	41.44	1.01	16.65	25.52
K-means with Forgy init.	41.45	6178.04	40.02	0.97	10.01	24.37
K-means with Mac Queen init.	43.41	6829.82	41.91	1.05	19.13	28.85
K-means with Kaufman init.	40.61	6011.59	39.21	0.89	9.52	27.59
K-means with Refinement init.	41.69	6212.71	40.25	0.97	9.12	17.24
K-means with MDC init.	45.46	7572.06	43.88	1.11	31.78	32.18
Single Linkage	50.28	7206.18	48.54	0.69	13.17	13.79
Complete Linkage	43.27	5894.69	41.78	1.25	5.54	16.09
Centroid Linkage	47.33	6713.86	45.70	0.78	9.27	11.49
Average Linkage	45.53	6498.85	43.96	0.85	6.68	11.49
Fuzzy C-Means	45.46	7572.06	43.89	1.11	31.50	28.74
K-means with imprv Pillar algorithm	40.64	5999.88	39.24	0.94	8.76	25.29

TABLE III
VALIDITY MEASUREMENT IN IRIS DATA SET

Algorithm	v_w	v_b	*SSE*	*SDVI*	*CPI*	*Error(%)*
K-means with random init.	0.59	181.47	0.58	4.81	3.83	14.73
K-means with Forgy init.	0.76	341.69	0.74	11.24	2.30	14.4
K-means with Mac Queen init.	0.62	318.32	0.61	7.80	1.25	14.47
K-means with Kaufman init.	0.54	302.23	0.53	5.85	0.51	11.33
K-means with Refinement init.	0.58	309.61	0.57	6.64	0.86	18
K-means with MDC init.	0.54	302.49	0.53	5.84	0.52	10.67
Single Linkage	0.97	317.98	0.95	4.34	2.54	32
Complete Linkage	0.61	300.84	0.60	5.22	0.86	16
Centroid Linkage	0.54	302.79	0.53	5.83	0.56	9.33
Average Linkage	0.54	302.79	0.53	5.83	0.56	9.33
Fuzzy C-Means	0.98	382.21	0.96	15.65	4.19	14.87
K-means with imprv Pillar algorithm	0.54	302.23	0.53	5.85	0.51	11.33

TABLE IV
VALIDITY MEASUREMENT IN NEW THYROID DATA SET

Algorithm	v_w	v_b	*SSE*	*SDVI*	*CPI*	*Error(%)*
K-means with random init.	136.40	19868.35	134.49	0.43	16.50	19.44
K-means with Forgy init.	136.85	22124.48	134.94	0.46	20.00	21.44
K-means with Mac Queen init.	136.41	20836.21	134.51	0.45	17.38	16.74
K-means with Kaufman init.	136.70	18581.93	134.80	0.44	16.62	14.88
K-means with Refinement init.	137.19	26366.74	135.28	0.48	26.30	30.51
K-means with MDC init.	137.73	27970.57	135.81	0.48	29.71	37.21
Single Linkage	278.14	119041.95	274.26	0.81	95.97	29.77
Complete Linkage	166.84	42480.35	164.51	0.40	54.85	28.37
Centroid Linkage	222.60	78800.70	219.49	0.17	87.38	27.91
Average Linkage	213.55	40171.26	210.57	0.27	61.19	26.05
Fuzzy C-Means	136.24	18621.09	134.34	0.43	13.73	14.42
K-means with imprv Pillar algorithm	136.21	18629.12	134.31	0.42	13.90	13.95

our improved Pillar algorithm and the other comparing algorithms. We only compared our algorithm to K-means algorithm with random initilization and K-means with several initialization algorithms which are Forgy, Mac Queen and MDC. We cannot compare with the rests of comparing algorithms because the execution process of those comparing algorithms got memory problem due to huge size of image pixel data. In Fig. 7, our improved Pillar

TABLE V
VALIDITY MEASUREMENT IN WINE DATA SET

Algorithm	v_w	v_b	*SSE*	*SDVI*	*CPI*	*Error(%)*
K-means with random init.	136.40	19868.35	134.49	0.43	16.50	33.20
K-means with Forgy init.	136.85	22124.48	134.94	0.46	20.00	32.36
K-means with Mac Queen init.	136.41	20836.21	134.51	0.45	17.38	37.42
K-means with Kaufman init.	136.70	18581.93	134.80	0.44	16.62	29.78
K-means with Refinement init.	137.19	26366.74	135.28	0.48	26.30	37.30
K-means with MDC init.	137.73	27970.57	135.81	0.48	29.71	42.70
Single Linkage	278.14	119041.95	274.26	0.81	95.97	57.30
Complete Linkage	166.84	42480.35	164.51	0.40	54.85	32.58
Centroid Linkage	222.60	78800.70	219.49	0.17	87.38	38.76
Average Linkage	213.55	40171.26	210.57	0.27	61.19	38.76
Fuzzy C-Means	136.24	18621.09	134.34	0.43	13.73	30.34
K-means with imprv Pillar algorithm	136.21	18629.12	134.31	0.42	13.90	29.78

TABLE VI
VALIDITY MEASUREMENT IN GLASS DATA SET

Algorithm	v_w	v_b	*SSE*	*SDVI*	*CPI*	*Error(%)*
K-means with random init.	136.40	19868.35	134.49	0.43	16.50	49.44
K-means with Forgy init.	136.85	22124.48	134.94	0.46	20.00	49.44
K-means with Mac Queen init.	136.41	20836.21	134.51	0.45	17.38	47.80
K-means with Kaufman init.	136.70	18581.93	134.80	0.44	16.62	46.26
K-means with Refinement init.	137.19	26366.74	135.28	0.48	26.30	49.39
K-means with MDC init.	137.73	27970.57	135.81	0.48	29.71	48.60
Single Linkage	278.14	119041.95	274.26	0.81	95.97	63.55
Complete Linkage	166.84	42480.35	164.51	0.40	54.85	51.40
Centroid Linkage	222.60	78800.70	219.49	0.17	87.38	63.08
Average Linkage	213.55	40171.26	210.57	0.27	61.19	62.15
Fuzzy C-Means	136.24	18621.09	134.34	0.43	13.73	46.12
K-means with imprv Pillar algorithm	1.80	272.41	1.75	31.49	13.92	48.13

TABLE VII
VALIDITY MEASUREMENT IN HEART DATA SET

Algorithm	v_w	v_b	*SSE*	*SDVI*	*CPI*	*Error(%)*
K-means with random init.	136.40	19868.35	134.49	0.43	16.50	41.89
K-means with Forgy init.	136.85	22124.48	134.94	0.46	20.00	41.89
K-means with Mac Queen init.	136.41	20836.21	134.51	0.45	17.38	42.67
K-means with Kaufman init.	136.70	18581.93	134.80	0.44	16.62	41.67
K-means with Refinement init.	137.19	26366.74	135.28	0.48	26.30	44.89
K-means with MDC init.	137.73	27970.57	135.81	0.48	29.71	42.22
Single Linkage	278.14	119041.95	274.26	0.81	95.97	46.11
Complete Linkage	166.84	42480.35	164.51	0.40	54.85	46.11
Centroid Linkage	222.60	78800.70	219.49	0.17	87.38	46.67
Average Linkage	213.55	40171.26	210.57	0.27	61.19	46.67
Fuzzy C-Means	136.24	18621.09	134.34	0.43	13.73	41.11
K-means with imprv Pillar algorithm	1.80	272.41	1.75	31.49	13.92	41.67

TABLE VIII
VALIDITY MEASUREMENT IN IONOSPHERE DATA SET

Algorithm	v_w	v_b	*SSE*	*SDVI*	*CPI*	*Error(%)*
K-means with random init.	6.93	829.40	6.89	0.49	2.39	28.92
K-means with Forgy init.	6.93	829.00	6.89	0.49	2.38	28.86
K-means with Mac Queen init.	8.06	1828.71	8.01	0.42	4.09	32.28
K-means with Kaufman init.	6.93	828.60	6.89	0.49	2.38	29.06
K-means with Refinement init.	6.93	829.40	6.89	0.49	2.39	28.77
K-means with MDC init.	6.93	829.40	6.89	0.49	2.39	28.77
Single Linkage	9.19	3205.77	9.14	0.33	6.26	35.61
Complete Linkage	8.83	351.26	8.78	0.66	1.68	35.33
Centroid Linkage	9.19	3205.77	9.14	0.33	6.26	35.61
Average Linkage	9.19	3205.77	9.14	0.33	6.26	35.61
Fuzzy C-Means	6.93	830.08	6.89	0.49	2.39	29.06
K-means with imprv Pillar algorithm	6.93	828.60	6.89	0.49	2.38	29.06

algorithm and MDC are able to cluster the image data clearly.We also test the computational speed for image data with different image sizes. Fig. 8 shows the comparison of computational time with different size of image data. Our improved Pillar algorithm performed the execution time very fast comparing to the other algorithms and close to the

Table 9 shows the comparison of execution times for

TABLE IX

COMPARISON OF EXECUTION TIMES (IN MS) FOR INITIAL CENTROIDS OPTIMIZATION ALGORITHMS OF K-MEANS FOR EACH DATA SET

Algorithm	Ruspini	Fossil	Iris	New Thyroid	Wine	Glass	Heart	Ionosphere
K-means with random init.	1.56	3.12	12.48	17.16	14.04	14.04	7.8	12.48
K-means with Forgy init.	45.24	40.56	101.4	138.84	131.04	174.72	88.92	157.561
K-means with Mac Queen init.	49.92	40.56	117	145	138.84	193.44	104.52	182.52
K-means with Kaufman init.	7597.25	6208.84	18766.92	37611.84	25662.16	129761.63	10810.87	42931.48
K-means with Refinement init.	1215.25	1230.85	4054.47	12041.72	12072.92	21978.98	6344.56	18453.36
K-means with MDC init.	31.2	46.8	46.8	78	78	187.2	62.4	93.6
K-means with imprv Pillar algorithm	1.56	7.8	14.04	9.36	15.6	21.84	7.8	135.72

performance of K-means. In Fig. 9, the improved Pillar algorithm can reduce drastically the exponential execution time of previous Pillar algorithm in line with huge and complex data.

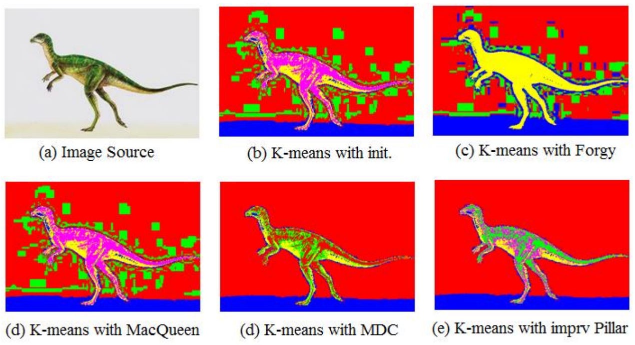

Fig. 7. visual comparisons of image clustering result

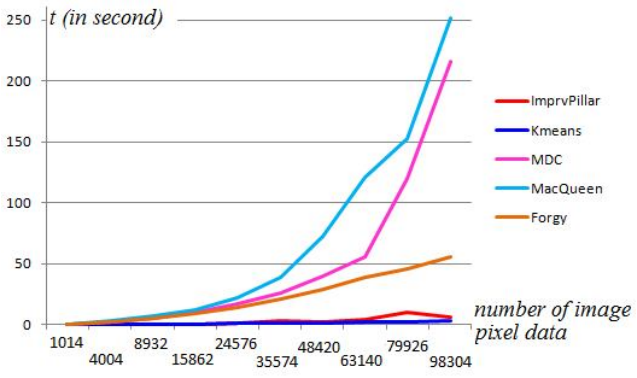

Fig. 8. comparison of computational time with different size of image data

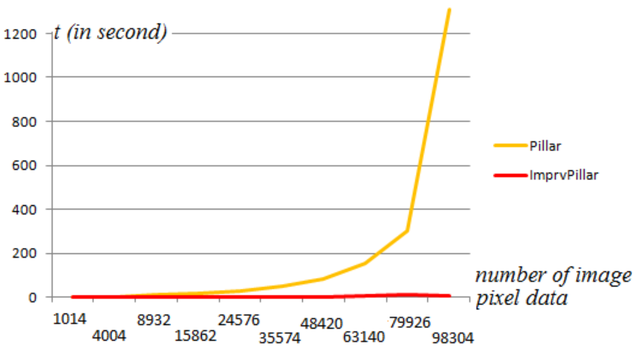

Fig. 9. Comparison time between our previous Pillar algorithm and current improved Pillar algorithm

VI. CONCLUSION

The Pillar algorithm is very effective to position the initial centroids for K-means clustering and improve the precision of the clustering results. However, the Pillar algorithm takes highly computational time for clustering huge data which often have many outliers. In this paper we present an improvement of our Pillar algorithm to speed up the computational time. We reduced the complexity of Pillar algorithm by excluding the initial centroids' neighbors for next steps of iterations. By this mechanism, the Pillar's complexity decreased from $O((k+h+1)\ n)$ to $O(n+(h_1.n_1)+\ldots+(h_k.n_k))$ in which number of involved data for distance calculation in next steps are reduced. A series of experiments performed that our improved Pillar algorithm can optimize the K-means clustering results quickly.

REFERENCES

[1] C. Yi-tsuu, "Interactive Pattern Recognition," Marcel Dekker Inc., New York and Basel, 1978.

[2] H. Ralambondrainy, "A conceptual version of the K-means algorithm," *Pattern Recognition Lett.*, 16, 1147-1157, 1995.

[3] P.S. Bradley, U.M Fayyad, "Refining initial points for K-means clustering," *Proc. 15th Internat. Conf. on Machine Learning* (ICML'98), 1998.

[4] G.A. Growe, "Comparing Algorithms and Clustering Data: Components of The Data Mining Process," thesis, department of Computer Science and Information Systems, Grand Valley State University, 1999.

[5] J.M. Penã, J.A. Lozano, P. Larrañaga, "An empirical comparison of the initilization methods for the K-means algorithm," *Pattern Recognition Lett.*, 20, 1027-1040, 1999.

[6] S. Ray, R.H. Turi, "Determination of number of clusters in K-means clustering and application in colthe image segmentation," *Proc. 4th ICAPRDT*, pp.137-143, 1999.

[7] B. Kövesi, J.M. Boucher, S. Saoudi, "Stochastic K-means algorithm for vector quantization," *Pattern Recognition Lett.*, 22, 603-610, 2001.

[8] M. Halkidi, Y. Batistakis, M. Vazirgiannis, "Clustering algorithms and validity measures," *Proc. 13th International Conference on Scientific and Statistical Database Management* (SSDBM'01), 2001.

[9] C.J. Veenman, M.J.T. Reinders, E. Backer, "A maximum variance cluster algorithm," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 9, pp. 1273-1280, 2002.

[10] V.E. Castro, "Why so many clustering algorithms-a position paper," *ACM SIGKDD Explorations Newsletter*, Vol. 4, Issue 1, pp. 65-75, 2002.

[11] Y.M. Cheung, "k*-Means: A new generalized k-means clustering algorithm," *Pattern Recognition Lett.*, 24, 2883-2893, 2003.

[12] A. R. Barakbah, K. Arai, "Identifying moving variance to make automatic clustering for normal dataset," *Proc. IECI Japan Workshop 2004* (IJW 2004), Musashi Institute of Technology, Tokyo, 2004.

[13] S.S. Khan, A. Ahmad, "Cluster center initialization algorithm for K-means clustering," *Pattern Recognition Lett.*, 25, 1293-1302, 2004.

[14] A.R. Barakbah, A. Helen, "Optimized K-means: an algorithm of initial centroids optimization for K-means," *Proc. Soft Computing, Intelligent System, and Information Technology* (SIIT) 2005, pp.2-63-66, Petra Christian University, Surabaya, 2005.

[15] A.R. Barakbah, A. Fariza, Y. Setiowati, "Optimization of Initial Centroids for K-means using Simulated Annealing," *Proc. Industrial Electronics Seminar* (IES) 2005, pp.286-289, Electronic Engineering Polytechnic Institute of Surabaya-ITS, Surabaya, 2005.

[16] A.R. Barakbah, "A new algorithm for optimization of K-means clustering with determining maximum distance between centroids," *Proc. Industrial Electronics Seminar* (IES) 2006, pp.240-244, Electronic Engineering Polytechnic Institute of Surabaya-ITS, Surabaya, 2006.

[17] A.R. Barakbah, K. Arai, "Hierarchical K-means: an algorithm for centroids initialization for K-means," *Reports of the Faculty of Science and Engineering*, Saga University, Japan, Vol. 36, No. 1, 2007.

[18] UCI Repository (http://www.sgi.com/tech/mlc/db/)

[19] A.R. Barakbah, Y. Kiyoki, "A Pillar Algorithm for K-Means Optimization by Distance Maximization for Initial Centroid Designation," *The IEEE Symposium on Computational Intelligence and Data Mining*, Nashville, 2009.

[20] J.Z. Wang, J. Li, G. Wiederhold, "Simplicity: Semantics-sensitive integrated matching for picture libraries", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23 (9), pp. 947–963, 2001.