k-Attractors: A Partitional Clustering Algorithm for Numeric Data Analysis

Y. Kanellopoulos*, P. Antonellis**, C. Tjortjis***, C. Makris**, N. Tsirakis**

*Software Improvement Group, 1070 NC

**Computer Engineering & Informatics Dept.University of Patras, 26500 Greece

***Dept. of Computer Science, University of Ioannina & Dept. Engineering Informatics and Telecommunications, University of Western Macedonia, 50100 Greece

y.kanellopoulos@sig.nl

{adonel, makri, tsirakis}@ceid.upatras.gr

Christos.Tjortjis@manchester.ac.uk

k-Attractors: A novel clustering algorithm

Yiannis Kanellopoulos

Software Improvement Group

A.J. Ernststraat 595-H

Amsterdam, The Netherlands

1082 LD

Abstract

Clustering is a data analysis technique, particularly useful when there are many dimensions and little prior information about the data. Partitional clustering algorithms are efficient, but suffer from sensitivity to the initial partition and noise. We propose here k-Attractors, a partitional clustering algorithm tailored to numeric data analysis. As a pre-processing (initialization) step, it employs maximal frequent itemset discovery and partitioning to define the number of clusters *k* and the initial cluster "attractors". During its main phase the algorithm utilizes a distance measure, which is adapted with high precision to the way initial attractors are determined. We applied k-Attractors as well as k-Means, EM and FarthestFirst clustering algorithms to several datasets and compared results. Comparison favored k-Attractors in terms of convergence speed and cluster formation quality in most cases, as it outperforms these 3 algorithms except from cases of datasets with very small cardinality containing only a few frequent itemsets. On the downside, its initialization phase adds an overhead that can be deemed acceptable only when it contributes significantly to the algorithm's accuracy.

1. Introduction

Clustering is a descriptive data mining task that aims to identify homogeneous groups of objects, based on the values of their attributes. It is particularly useful in problems where there is little prior information available about the data, and a minimum number of assumptions are required. Clustering is appropriate for the exploration of interrelationships among the data points when assessing their structure (Jain et al., 1999).

Clusters can be broadly created by employing either hierarchical or partitional algorithms. The former organize data into a hierarchical structure based on a proximity matrix, the latter identify the partition that optimizes, usually locally, a clustering criterion.

k-Means is a classic partitional clustering algorithm (Hartigan 1975). It represents each cluster with the mean value of its objects. As a result, inter-cluster similarity is measured based on the distance between the object and the mean value of the input data in a cluster. It is an iterative algorithm in which objects are moved among clusters until a desired set is reached. Its main problems are that the users have to define the number of clusters *k*, and that it is sensitive to the initial partitioning. That is, different initial partitions, indicated by user input, lead to different results (Han, Kamber, 2001).

This paper presents a further elaborated version of k-Attractors, a partitional clustering algorithm introduced in (Kanellopoulos et al. 2007), which has the following characteristics:

- It locates the initial attractors of cluster centers with high precision.
- The final number of the derived clusters is defined without user intervention, by using the maximal frequent itemset discovery.
- It measures distance based on a composite metric that combines the Hamming distance and the inner product of transactions and clusters' attractors.

The k-Attractors algorithm can be employed for numeric data analysis and is based on the assumption that such data demonstrate identifiable patterns. The most prevalent of these patterns, that is the initial centers and the number of clusters, can form the basis for cluster analysis. For this reason we employ the frequent itemsets discovery technique, which has the ability to extract such patterns from large datasets (Agarwal and Srikant, 1994).

WEKA's implementations of k-Means, EM and FarthestFirst algorithms (Witten, Frank, 2005) were used in order to compare our algorithm in terms of performance (i.e. convergence speed) and accuracy (i.e.

quality clusters formation). The results are promising, as k-Attractors in its main phase, was faster and formed more accurate (i.e. better quality) clusters than the above mentioned algorithms, in most cases. The algorithm did not perform well in cases where the data did not exhibit a sufficient number of patterns that could be captured. Also, the experiments showed that the initialization phase of k-Attractors adds an overhead which can be acceptable only in the cases that it contributes significantly to algorithm's accuracy.

The remaining of the paper is organized as follows: section 2 presents a review of existing clustering algorithms, section 3 discusses the background on clustering, association rules and related problems, section 4 introduces our approach, section 5 details k-Attractors, section 6 presents experimental results and section 7 concludes with directions for future work.

2. Clustering Algorithms Review

Various research works have been reported to deal with clustering algorithms. Zhuang and Dai H. present a Maximal Frequent Itemset Approach for clustering web documents. Based on maximal frequent itemset discovery, they propose an efficient way to precisely locate the initial points for the k-Means algorithm (Hartigan 1975). Fung et al. (Fung, Wang, Ester, 2003) propose the Frequent Itemset-based Hierarchical Clustering (FIHC) for document clustering. The intuition of this algorithm is that there are some frequent itemsets for each cluster (topic) in the document set, and different clusters share few frequent itemsets. Wang et al. (Wang, Xu, Liu, 1999) propose a similarity measure for a cluster of transactions based on the notion of large items. Han et al. (Han et al., 1997) introduced Association Rules Hypergraph Partitioning (ARHP) algorithm. It constructs a weighted hypergraph to represent the relationships among discovered frequent itemsets. It aims to find *k* partitions such that the vertices in each partition are highly related. Kosters et al. (Kosters et al., 1999) propose a method that employs association rules having high confidence to construct a hierarchical sequence of clusters. The work of Li et al. (Li et al., 2007)

discusses an efficient method based on spectra analysis in order to effectively estimate the number of clusters in a given data set.

The work of Jing et al. (Jing, Ng and Zhexue, 2007) provides a new clustering algorithm called EWKM which is a k-means type subspace clustering algorithm for high-dimensional sparse data. Patrikainen and Meila present a framework for comparing subspace clusterings (Patrikainen and Meila 2006), while Cheng et al. introduce a new algorithm called dual clustering on two domains, the optimization and the constraint domain (Lin et. al., 2005). This algorithm combines the information in both domains and iteratively uses a clustering algorithm on the optimization domain and also a classification algorithm on the constraint domain to reach the target clustering effectively.

Except from the algorithms described above, three of the most important clustering algorithms are k-Means, EM and Farthest First (Jain et al., 1999), (Witten, Frank, 2005). They are characteristic, commonly used clustering algorithms. Moreover, they are implemented in Weka 3 (Weka), an open source data mining tool frequently used in the literature, thus facilitating experimental replication. Those algorithms are extensively described in (Witten, Frank, 2005).

k-means is a classic iterative-based clustering algorithm. The user firstly specifies in advance how many clusters are being sought: this is the parameter *k*. Then *k* points are chosen at random as clusters centers. All instances are assigned to their closest cluster center according to the ordinary Euclidean distance metric. Next the centroid, or mean, of the instances in each cluster is calculated. These centroids are taken to be new center values for their respective clusters. Finally, the whole process is repeated with the new cluster centers. Iteration continues until no more points change clusters, at which stage the cluster centers are stable (Witten, Frank, 2005).

The EM (Expectation Maximization) clustering algorithm is based on a statistical model called finite mixtures. A mixture is a set of *k* probability distributions, representing *k* clusters, that govern the attribute values for members of that cluster. In other words, each distribution gives the probability that a particular

instance would have a certain set of attribute values if it was known to be a member of that cluster. Each cluster has a different distribution. Finally, the clusters are not equally likely: there is some probability distribution that reflects their relative populations. The problem is that we know neither the distribution that each training instance came from, nor the parameters of the mixture model. For that reason, the EM adopts the same procedure used for the k-means clustering algorithm and iterates. It starts with some initial guesses for the parameters of the mixture model; it uses them to calculate the cluster probabilities for each instance, and finally uses these probabilities to re-estimate the parameters and repeats (Witten, Frank, 2005).

The FarthestFirst algorithm performs a farthest-first traversal over the dataset. Starting with an arbitrary chosen point and adding it to the set, the algorithm picks that point in the dataset which is farthest away from the current centroids and adds to X in each iteration.

At the end of *k* iterations, each point in X acts as a center for a stratum and the result of the algorithm is a disjoint partitioning, obtained by assigning each point in the dataset to its nearest point in X (Witten, Frank, 2005).

In section 6 we base our experiments on comparing k-Attractors with the three algorithms described above. This facilitates repeatability of the experiments as Weka is an open source data mining suite.

3. Preliminaries

This section presents the main concepts that form the basis for k-Attractors. More specifically, we present the concepts of the frequent itemsets discovery and their subsequent graph construction and partitioning. These two techniques help creating the initial attractors and defining the maximum number *k* of the derived clusters.

k-Attractors Algorithm

/*Input Parameters*/

Support: s

Hamming distance power: h

Inner product power: i

Given a set of m data items $t_1, t_2, \ldots, t_m$

/*Initialization Phase*/

(1) Generate frequent itemsets using the APriori Algorithm;

(2) Construct the *itemset graph* and partition it using the confidence similarity criteria related to the support of these itemsets;

(3) Use the number of partitions as the final k;

(4) Select the maximal frequent itemset of every cluster in order to form a set of k initial attractors;

/*Main Phase*/

Repeat

(6) Assign each data item to the cluster that has the minimum $Score(C_i \rightarrow t_j)$;

(7) When all data items have been assigned, recalculate the new attractors;

Until t_i don't_move

(8) Search all clusters to find outliers and group them in a new cluster

Figure 1 - k-Attractors overview

3.1 Frequent Itemsets Discovery– Apriori Algorithm

A frequent itemset is a set of items that appear together in more than a minimum fraction of the whole dataset.

More specifically let J be a set of quantitative data. Any subset I of J is called an itemset. Let T = <t1,...,tn> be a sequence of itemsets called a transaction database. Its elements t ϵ T will be called itemsets or transactions.

An itemset can be frequent if its support is greater than a minimum support threshold, denoted as min_sup. The support of an item X in T denoted as min_sup T(X) is the number of transactions in T that contain X. The term frequent item refers to an item that belongs to a frequent itemset. If now, an item X is frequent and no superset of X is frequent, then X is a maximally frequent itemset; and we denote the set of all maximally frequent itemsets by MFI. From these definitions it is easy to see that the following relationship holds MFI $\subseteq$ FI, where FI is the set of the most frequent itemsets. Apriori, the well known association rule mining algorithm, was used in order to discover frequent itemsets (Agarwal and Srikant, 1994). According to (Bodon, 2005) the Apriori's moderate traversal of a data search space is the most suitable when mining very large databases. Apriori scans the transaction dataset several times. After the first scan, the frequent items are found, and in general after the l^{th} scan, the frequent item sequences of size l (we call them l-sequences) are extracted. The method does not determine the support of every possible sequence. In an attempt to narrow the domain to be searched, before every pass it generates candidate sequences. A sequence becomes a candidate if every subsequence of it is frequent. Obviously every frequent sequence is a candidate too; hence it is enough to calculate the support of candidates. Frequent l-sequences generate the candidate (l + 1)-sequences after the l^{th} scan. Candidates are generated

in two steps. First, pairs of l-sequences are found, where the elements of the pairs have the same prefix of size l−1. Here we denote the elements of such a pair with <i1, i2,..., il−1, il> and <i1, i2,..., il−1, i'l>.

Depending on items il and i'l we generate one or two potential candidates. If il <> il' then they are <i1, i2,..., il−1, il, i'l> and <i1, i2,..., il−1, i'l, il>, otherwise it is <i1, i2,..., il−1, il, il> (Manilla et. al. 1995). In the second step the l-subsequences of the potential candidate are checked. If all subsequences are frequent, it becomes a candidate. As soon as the candidate (l + 1)-sequences have been generated, a new scan of the transactions is started and the precise support of the candidates is determined. This is done by reading the transactions one-by-one. For each transaction *t* the algorithm decides which candidates are contained by *t*. After the last transaction is processed, the candidates with support below the support threshold are thrown away. The algorithm ends when no new candidates are generated.

3.2 Frequent Itemsets Graph Partitioning – kMetis Algorithm

The discovered frequent itemsets often reveal hidden relationships and correlations among data items. By constructing and partitioning a weighted hypergraph from those itemsets, using similarity (i.e. based on hyperdges' weights) criteria related to the confidence of their association rules, we can reduce their size by eliminating those that are not similar.

A hypergraph H = (V, E) consists of a set of vertices (V) and a set of hyperedges (E). A hypergraph is an extension of a graph in the sense that each hyperedge can connect more than two vertices. In this work, the vertex set corresponds to the distinct items in the database and the hyperedges correspond to the frequent itemsets. For example, if {A B C} is a frequent item-set, then the hypergraph contains a hyperedge that connects A, B, and C. The weight of a hyperedge is determined by a function of the confidences for all the association rules involving all the items of the hyperedge.

The kMetis algorithm (Karypis and Kumar, 1998) has a very simple structure. The graph G = (V, E) is first coarsened down to a small number of vertices, a k-way partitioning of this much smaller graph is

computed, and then this partitioning is projected back, towards the original graph (finer graph) by successively refining the partitioning at each intermediate level.

4. Our approach

In this research work we propose a partitional algorithm that utilizes a preprocessing method for its initial partitioning and incorporates a distance measure adapted to the way initial attractors are determined by this method.

More specifically, k-Attractors employs the maximal frequent itemset discovery and partitioning, similarly to (Zhuang, Dai, 2004), (Han et al., 1997). However, k-Attractors is different as it is not used in the context of document clustering. It is applied to numeric data that are expected to demonstrate patterns that can be identified by the maximal frequent itemsets discovery technique. The discovered frequent itemsets often reveal hidden relationships and correlations among data items. By constructing and partitioning a weighted hypergraph from those itemsets, using similarity criteria (based on the weights of hyperedges) related to the confidence of their association rules, their size can be reduced by eliminating those that are not interesting (i.e. their support is below a certain threshold). The final set of rules will be used in order to define the final number of clusters and the initial attractors of their centers. The intuition in k-Attractors is that a frequent itemset is a set of measurements that occur together in a minimum part of a dataset. Transactions with similar measurements are expected to be in the same cluster. The term attractor is used instead of centroid, as it is not determined randomly, but is determined by its frequency in the whole population of a given dataset.

An important characteristic of k-Attractors is that it proposes a distance measure which is adapted to the way the initial attractors are determined by the preprocessing method. Hence, it is primarily based on the comparison of frequent itemsets. More specifically, a composite metric based on the Hamming distance and the dot (inner) product between each transaction and the attractors of each cluster is utilized.

The Hamming distance is given by the number of positions that a pair of strings differ from each other. Put another way, it measures the number of substitutions required to transform the one string to the other. In this work we consider a string as a set of data items, and more specifically as a vector of numeric data. Furthermore, the dot product of two vectors is a measure of the angle and the orthogonality of two vectors. It is used in order to compensate for the position of both vectors in the Euclidean space (Kosters et al., 1999).

5. Algorithm Description

This section details the basic steps of k-Attractors along with some examples.

5.1 Overview

The two basic phases of k-Attractors are:

- Initialization phase:
    - The first step of this phase generates frequent itemsets using the APriori algorithm. The derived frequent itemsets are used to construct the itemset graph, and kMetis, a graph partitioning algorithm, is used to find the maximum number of the desired clusters and to assign each frequent itemset to the appropriate cluster.
    - As soon as the maximum number of the desired clusters *k* is determined, we select the maximal frequent itemsets from every cluster, forming a set of *k* frequent itemsets as the initial attractors.
- Main phase:
    - As soon as the attractors are found, we assign each transaction to the cluster that has the minimum distance against its attractor.

- When all transactions have been assigned to clusters we recalculate the attractors for each cluster, in the same way as in the initialization phase.

Figure 1 illustrates an overview of k-Attractors. We detail its two phases next.

5.2 Initialization Phase

The goal of the initialization phase is two-fold: firstly to identify the most frequent itemsets of the input data and secondly to determine the number of clusters.

In order for the most frequent itemsets to be discovered, we apply the APriori algorithm against the input data file. The APriori algorithm takes as input the absolute support s of the required itemsets and returns all the one-dimensional and multi-dimensional itemsets with support greater than or equal to s. The value of s is defined empirically by an expert user, based on the statistical characteristics of the dataset under consideration.

Once the most frequent itemsets have been discovered, we form the itemset graph. Given the set of the most frequent itemsets $FI = \{fi_1, fi_2, \ldots fi_m\}$, the itemset graph is a graph $G(V, E)$, where $V = \{fi_i \in FI\}$ and $E = \{e_{ij} | fi_i \cap fi_j \neq \varnothing\}$. The intuition behind this graph is that if two itemsets have at least one common item, then they should possibly belong to the same cluster and thus we connect them with an edge in the itemset graph. For more accuracy, we could have weighted each edge with the number of common items between the two corresponding itemsets/vertices, but in order to keep the initialization phase as simple and fast as possible we decided not to weigh the edges.

Figure ***2*** demonstrates an example of the itemset graph's construction.

In the next step we apply a graph partitioning algorithm to the itemset graph. In our case, we utilized the kMetis algorithm in order to partition the itemset graph (Han et al., 1997). kMetis partitions the itemset

graph into a number of distinct partitions and assigns each vertex of the graph (i.e. each itemset) to a single partition.

The final number of the derived partitions is the number of clusters that we use in the main phase of the k-Attractors algorithm.

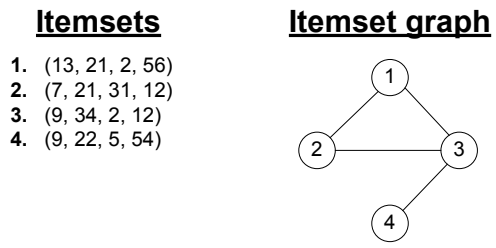

Figure 2. Itemset graph example

The final step of the initialization phase is the attractors' discovery. During this step, every previously determined graph partition is examined. In (Kanellopoulos et al. 2007) the procedure for discovering the initial attractors is summarised. For each partition we find the maximal frequent itemset MI belonging to this partition, and check the cardinality of its dimensions. If the number of dimensions is equal to the input data items' number of dimensions n, then we assign the corresponding itemset as the attractor of the corresponding partition. However, in most cases, the cardinality of the maximal itemset is less than n. In such cases, we search for the next maximal frequent itemset NI in the corresponding partition and merge it with the previous itemset MI. Merging occurs only in dimensions that are absent from the first maximal itemset MI. We repeat this procedure until we have formed an itemset with cardinality equal to n, and assign the formed itemset as attractor of the corresponding partition.

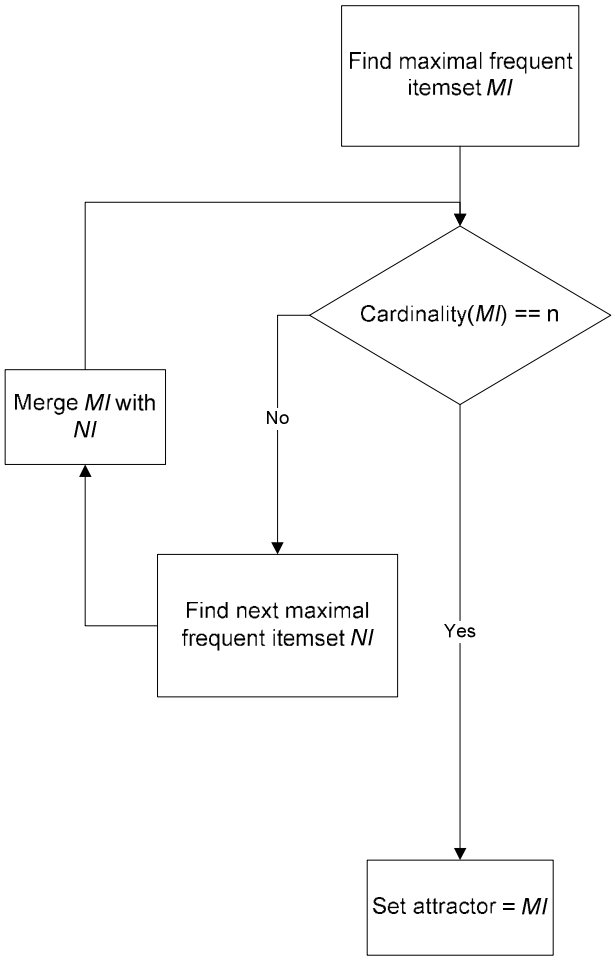

Figure 3 Attractors discovery

In order to provide more flexibility, k-Attractors performs a post-processing step against the previously determined attractors. The algorithm defines a threshold, called attractor similarity ratio, r. This threshold defines the maximum allowed similarity between two different attractors. After several experiments for calibrating k-Attractors we use 0.8 as the default value for r. The similarity between two attractors a_1, a_2 is defined as follows:

$$sim(a_1, a_2) = \frac{\#(a_1 \cap a_2)}{n} \qquad (1)$$

In other words, the similarity between two attractors is the ratio of the number of common items divided by the number of total dimensions.

If the similarity of two attractors is more than r, then we randomly discard one of the two attractors; thus the number of total partitions is decreased by one.

It has to be noted that the initialization phase adds a significant overhead to the overall algorithm's performance. The reason for this is the combination of the APriori and kMetis algorithms for the extraction of frequent itemsets and their partitioning, respectively. As long as it contributes significantly to the algorithm's accuracy comparing to other algorithms this overhead is not considered a disadvantage.

5.3 Main Phase

The goal of k-Attractor's main phase is to assign each input data item to a cluster, using a partitional approach.

At first, we form a set of *k* empty clusters, where *k* is the number of distinct graph partitions discovered during the initialization phase. Every formed cluster is then assigned to the corresponding attractor calculated in the previous phase.

Once the clusters have been formed, the main phase of k-Attractors begins. This phase resembles a partitional algorithm, where every data item is assigned to a cluster according to a predefined distance metric and in every step the centers of every cluster are re-calculated until the algorithm converges to a stable state where the clusters' centers do not change.

The k-Attractors algorithm utilizes a hybrid distance metric based on vector representation of both the data items and the cluster's attractors. The distance of these vectors is measured employing the following composite distance metric:

$$Score\left(C_i \leftarrow t_j\right) = h * H\left(a_i, t_j\right) + i * \left(a_1 t_1 + \ldots a_n t_n\right) \qquad (2)$$

In this formula, the first term is the Hamming distance between the attractor a_i and the data item t_j. It is given by the number of positions that a pair of strings is different and is defined as follows:

$$H\left(a_i, t_j\right) = n - \#\left(a_i \cap t_j\right) \qquad (3)$$

As our algorithm is primarily based on itemset similarity, we want to measure the number of substitutions required to change one itemset into another. The second term is the dot (inner) product between this data item and the attractor a_i. It is used in order to compensate for the position of both vectors in the Euclidean space (Kosters et al., 1999).

The multipliers h, i in equation (2) define the metric's sensitivity to Hamming distance and inner product respectively. For example, $i = 0$ would indicate that the composite metric is insensitive to the inner product between the data item and the cluster's centroid. Both h and i are taken as input parameters in our algorithm during its execution. Thus, k-Attractors provides the flexibility of changing the sensitivity of the composite distance metric to both Hamming distance and inner product.

Utilizing this distance metric, k-Attractors assigns every data item to a single cluster and recalculates the attractors for every cluster. Recalculation includes finding the data item in a cluster that minimizes the

total sum of distances between the other data items belonging to that cluster. The above procedure is repeated until the attractors of all clusters do not change any further.

The final step of the k-Attractors' main phase involves outlier handling. During this step, every cluster is checked for outliers, according to a threshold d. Specifically, a data item t, belonging to cluster i, is considered an outlier if:

$$Score(a_i, t) >= d \cdot avg_i\left(Score(a_i, t_j)\right) \quad (4)$$

where $avg_i\left(Score(a_i, t_j)\right)$ is the average distance between the data items of cluster i and the attractor a_i.

Dataset	# Items	# Classes
Iris	150	3
Wisconsin Breast Cancer	699	2
Vehicle	946	4
Haberman Surgery	306	2
Software Measurement Data	50	4

Table 1: Characteristics of the employed datasets

The discovered outliers of all clusters are grouped into a new cluster, called outliers cluster, thus the total number of formed clusters is k+1.

6. Experimental Results

6.1 Datasets used

The evaluation of the proposed clustering algorithm involved two main experiments that were performed in a Pentium M 2.0 GHz machine with 1GByte RAM. The scope of those experiments was the comparison of k-Attractors with the set of clustering algorithms contained in WEKA 3 (Weka), the Java open source data mining tool. In particular, we compared k-Attractors with the clustering algorithms described in Section 2: k-Means, EM and FarthestFirst.

The datasets used in our experiments were the Iris dataset (Orange), the Wisconsin Breast Cancer dataset (Orange), the Vehicle dataset (Orange), the Haberman Surgery dataset (Machine Learning Datasets) and an industrial software measurement dataset derived from parsing a fragment of System2, a large logistics system implemented in Java (6782 classes). This last dataset consisted of a set of 10 calculated software metrics for 50 java classes of the parsed system.

The main characteristics of the employed datasets are summarized in Table 1.

6.2 k-Attractors evaluation

The purpose of our experiments was to evaluate k-Attractors against the clustering algorithms contained in WEKA 3: k-Means, EM and FarthestFirst. For that reason, we used 5 datasets for which there is previous knowledge of data item classification.

For our experiments, we used the recall, precision and F-measure external metrics in order to evaluate the accuracy (quality) of the clusters formed by k-Attractors, k-Means, EM and FarthestFirst for every utilized dataset. Precision, recall and F-measure are external cluster quality metrics based on the comparison of the formed clusters to previously known external classes (e.g. different domains). Given a

class Z_j of data items with a number n_j of items and a cluster C_i, formed by a clustering algorithm, with n_i items, let n_i^j be the number of items in C_i belonging to Z_i. Then, the precision, recall and F-measure are defined as follows:

$$\text{precision}(C_i, Z_j) = \frac{n_i^j}{n_i} \quad (5)$$

$$\text{recall}(C_i, Z_j) = \frac{n_i^j}{n_j} \quad (6)$$

$$\text{F-measure}(C_i, Z_j) = \frac{\text{precision}(C_i, Z_j) \times \text{recall}(C_i, Z_j)}{\text{precision}(C_i, Z_j) + \text{recall}(C_i, Z_j)} \quad (7)$$

Cluster	Popul.	k-Attractors			k-Means		
		Precision	Recall	F-measure	Precision	Recall	F-measure
1	458	0,98	0,97	0,98	0,96	0,92	0,94
2	241	0,95	0,97	0,96	0,96	0,98	0,97
		Weighted F-measure		**0,97**	**Weighted F-measure**		**0,95**
		Iterations		**4**	**Iterations**		**5**

Table 2. Breast cancer experimental results (k-Attractors and k-Means)

Cluster	Popul.	EM			FarthestFirst		
		Precision	Recall	F-measure	Precision	Recall	F-measure
1	458	1,00	0,92	0,96	0,96	0,93	0,94
2	241	0,87	1,00	0,93	0,96	0,98	0,97
		Weighted F-measure		**0,92**	**Weighted F-measure**		**0,95**

Table 3. Breast cancer experimental (EM and FarthestFirst)

Cluster	Popul.	k-Attractors			k-Means		
		Precision	Recall	F-measure	Precision	Recall	F-measure
1	50	0.98	1.00	0.99	0,77	0.94	0.85

2	50	0.92	0.68	0.78	1.00	1.00	1.00
3	50	0.76	0.94	0.84	0.92	0.72	0.81
		Average F-measure		**0.87**	**Average F-measure**		**0.89**
		Iterations		**20**	**Iterations**		**4**

Table 4. Iris experimental results (k-Attractors and k-Means)

Cluster	**Popul.**	**EM**			**FarthestFirst**		
		Precision	**Recall**	**F-measure**	**Precision**	**Recall**	**F-measure**
1	50	0,78	0,90	0,88	0,85	0,70	0,77
2	50	1,0	1,0	1,0	1,0	1,0	1,0
3	50	1,0	0,72	0,84	0,75	0,88	0,81
		Weighted F-measure		**0,90**	**Weighted F-measure**		**0,86**

Table 5. Iris experimental results (EM and FarthestFirst)

Cluster	**Popul.**	**k-Attractors**			**k-Means**		
		Precision	**Recall**	**F-measure**	**Precision**	**Recall**	**F-measure**
1	240	0.43	0.35	0.38	0.41	0.47	0.44
2	240	0.59	0.52	0.55	0.42	0.33	0.37
3	240	0.37	0.38	0.38	0.23	0.21	0.22
4	226	0.36	0.45	0.40	0.42	0.49	0.45
		Weighted F-measure		**0,43**	**Weighted F-measure**		**0,37**
		Iterations		**2**	**Iterations**		**8**

Table 6. Vehicle experimental results (k-Attractors and k-Means)

Cluster	**Popul.**	**EM**			**FarthestFirst**		
		Precision	**Recall**	**F-measure**	**Precision**	**Recall**	**F-measure**
1	240	0,41	0,47	0,44	0,27	0,77	0,39
2	240	0,42	0,33	0,33	0,49	0,21	0,29
3	240	0,23	0,21	0,21	0,33	0,01	0,02
4	226	0,42	0,49	0,49	0,35	0,21	0,26
		Weighted F-measure		**0,37**	**Weighted F-measure**		**0,24**

Table 7. Vehicle experimental results (EM and FarthestFirst)

Cluster	Popul.	k-Attractors			k-Means		
		Precision	Recall	F-measure	Precision	Recall	F-measure
1	225	0,56	0,77	0,65	0,28	0,57	0,37
2	81	0,91	0,78	0,84	0,75	0,47	0,58
		Weighted F-measure		**0,69**	**Weighted F-measure**		**0,43**
		Iterations		**2**	**Iterations**		**6**

Table 8. Haberman Surgery experimental results (k-Attractors and k-Means)

Cluster	Popul.	EM			FarthestFirst		
		Precision	Recall	F-measure	Precision	Recall	F-measure
1	225	0,42	0,67	0,51	0,74	0,98	0,84
2	81	0,85	0,67	0,75	0,33	0,02	0,05
		Weighted F-measure		**0,57**	**Weighted F-measure**		**0,63**

Table 9. Haberman Surgery experimental results (EM and FarthestFirst)

Cluster	Popul.	k-Attractors			k-Means		
		Precision	Recall	F-measure	Precision	Recall	F-measure
1	35	0.88	0.97	0.92	0.80	0.65	0.72
2	9	0.88	0.80	0.84	0.15	0.50	0.23
3	4	0.66	0.66	0.66	1.00	0.70	0.82
4	2	0.50	0.25	0.33	0.00	0.00	0.00
		Weighted F-measure		**0,85**	**Average F-measure**		**0,60**

Table 10. Software measurement data experimental results

In order to properly measure the total quality of the formed clusters by every clustering algorithm, we calculated the weighted F-measure for the clusters formed by every clustering algorithm in each dataset and compared the resulted weighted F-measures. The weighted F-measure of a clustering algorithm in a dataset is defined as follows:

$$WF = \frac{\sum_{i=1}^{k}(n_i * F_i)}{n} \quad (8)$$

where F_i denotes the F-measure for the i-th cluster and n_i denotes the number of items in the i-th cluster.

Additionally, for every dataset used, we measure the number of iterations required by k-Attractors's main phase and k-Means in order for the clusters to converge into a stable state. As every iteration in k-Attractors's main phase and k-Means has a complexity of O(n) we can compare the performance of the two algorithms by comparing the number of iterations performed. The more the required iterations, the longer the clustering process lasts. However, because EM and FarthestFirst are not partitional clustering algorithms, we did not compare k-Attractors with them in terms of performance (i.e. converge speed).

Table 2 and Table 3 present the experimental results for the Breast Cancer dataset. As it can easily be seen, k-Attractors in its main phase performs slightly better than k-Means and FarthestFirst (weighted F-measure 0.97 regarding the weighted F-measure 0.95 of k-Means and FarthestFirst) and much better than EM. Additionally, it requires one less iteration (4 iterations regarding the 5 iterations of k-Means). Thus, regarding the Breast Cancer dataset, k-Attractors is better compared to the other 3 clustering algorithms in terms of both accuracy and performance.

Table 4 and Table 5 present the experimental results for the Iris dataset. It can be seen that k-Attractors in its main phase performs slightly worse than k-Means and EM (weighted F-measure 0.87 regarding the weighted F-measure 0.89 of k-Means and 0.90 of EM) and slightly better than FarthestFirst. In addition, the main phase of k-Attractors requires 20 iterations regarding the 4 iterations required by k-Means until it converges. This is due to the fact that the Iris dataset contains only a few frequent itemsets, with very small cardinality. Hence, k-Attractors cannot utilize those frequent itemsets in order to form the initial attractors for the clustering process. k-Attractors utilizes the statistical mean to approximate the value of each attribute in the initial attractors instead. This approach leads to the fact that the initial attractors are

very similar to each other, because most of their attributes have the same value and as a result the performance of k-Attractors is relatively poor.

Table 6 and Table 7 present the experimental results for the Vehicle dataset. As it can easily be seen, k-Attractors in its main phase performs better than the other clustering algorithms (weighted F-measure 0.43 regarding the weighted F-measure 0.37 of k-Means, 0.37 of EM and 0.24 of FarthestFirst). Additionally, k-Attractors requires only 2 iterations compared to the 8 iterations required by k-Means. This is due to the fact, that the Vehicle dataset contains enough frequent itemsets in order for the initial attractors to be calculated right, thus k-Attractors in its main phase performs better than the other clustering algorithms both in accuracy and time performance.

Table 8 and Table 9 present the experimental results for the Haberman surgery dataset. As it can be seen, k-Attractors in its main phase performs better than the other 3 algorithms and requires only 2 iterations while k-Means requires 6 iterations. This performance boost comes from the fact that the range of attribute values in this dataset is relatively small (varies between 1 and 5), thus there exist a lot of frequent itemsets in order to efficiently calculate the initial attractors for the k-Attractors algorithm.

Table 10 shows that k-Attractors' clusters are closer to the domain expert's clusters. k-Attractors achieves a weighted 0.85 F-measure, while k-Means achieves a weighted 0.60 F-measure. Especially, regarding the two largest clusters (cluster 1 and cluster 2), the corresponding calculated F-measure is very high and better than the two corresponding k-Means clusters. Considering now, the smallest cluster in k-Attractors (cluster 4) and k-Means (cluster 4), these clusters correspond to the domain expert's outlier cluster. It is obvious that k-Attractors approximates the domain expert's cluster because of the application of the outlier handling phase. k-Means lacks such a phase, thus the recall, precision and f-measure of the corresponding cluster are both 0.

Hence, the experimental results show that k-Attractors forms more accurate (better quality) clusters, in terms of higher F-Measure, than k-Means, approximating the domain expert's manually created clusters. This can be explained by the fact that software measurement data are expected to demonstrate specific

behavior and not random patterns or trends as in software development projects programmers follow certain specifications, design guidelines and code-styles. Thus, this type of data usually contains frequent common itemsets which can be captured during k-Attractors's initial phase leading to more accurate results (Kanellopoulos et al. 2007).

6.3 Discussion

We conducted a series of experiments in order to compare k-Attractors with kMeans, EM and FarthestFirst clustering algorithms. The results are promising as in most cases k-Attractors, in its main phase, performs better than the other clustering algorithms both in performance (converge-speed) and in accuracy (quality of the formed clusters). This efficiency of k-Attractors is a result of the calculation of the approximate initial attractors for each formed cluster during the initialization phase. However, this phase adds a significant overhead to the overall algorithm's performance but as long as it contributes significantly to its accuracy this overhead is considered justifiable.

K-Attractors performed worse on the Iris dataset compared to k-Means, EM and FarthestFirst in terms of performance and accuracy. This result was due to the fact that the Iris dataset contains only a few frequent itemsets with very small cardinality. Hence, k-Attractors could not utilize those frequent itemsets in order to form the initial attractors for the clustering process and was outperformed by the other clustering algorithms. As mentioned in section 5.2, in case of few discovered frequent itemsets, k-Attractors utilises the statistical mean of every dimension in order to calculate the attractors. Thus the formed attractors are very similar to each other resulting in poor performance. It would be interesting to utilise a more sophisticated statistical measurement in order to approximate the missing dimensions and test whether this improves the performance of k-Attractors.

7. Conclusions and Future Work

The aim of this work was the development of a new partitional clustering algorithm, k-Attractors, tailored to numeric data analysis, overcoming the weaknesses of other partitional algorithms.

The initialization phase of the proposed algorithm involves a preprocessing step which calculates the initial partitions for k-Attractors. During this phase, the exact number of k-Attractors clusters was calculated in addition with the initial attractors of each cluster. Thus the problems of defining the number of clusters and initializing the centroids of each cluster are resolved. In addition, the constructed initial attractors approximate the real clusters' attractors, improving that way the convergence speed of the proposed algorithm.

The main phase of k-Attractors forms clusters employing a composite distance metric which utilises the Hamming distance and the inner product of data item vector representations. Thus, the employed metric is adapted to the way the initial attractors are determined by the preprocessing step.

The last step deals with outliers and is based on the distance between a data item and its cluster's attractor. The discovered outliers are grouped into a separate cluster.

The results from the conducted experiments are promising, as k-Attractors' main phase outperformed in performance and accuracy the other algorithms in most of the cases. This is attributed to its initialization phase which however adds an overhead which is deemed acceptable when it contributes significantly to algorithm's accuracy.

For this reason we plan on improving the way the initial attractors are derived in order to minimize the cost of the initialization phase. We could also attempt to customise the proposed distance metric in order to adapt it to categorical semantics thus making it applicable to categorical datasets.

Acknowledgements

This research work has been partially supported by the Greek General Secretariat for Research and Technology (GSRT) and Dynacomp S.A. within the program "P.E.P. of Western Greece Act 3.4". We would also like to thank Rob van der Leek, Patrick Duin and Harro Stokman from the Software Improvement Group for their valuable comments and feedback concerning our clustering results.

References

[1] (Agarwal and Srikant, 1994) Agarwal R, and Srikant R. 1994. Fast Algorithms for Mining Association Rules in Large Databases, In Proceedings of 20th International Conference VLDB pp. 487-499.

[2] (Bodon, 2005) Bodon F. 2005. A Trie-based APRIORI Implementation for Mining Frequent Item sequences, OSDM.

[3] (Fung, Wang, Ester, 2003) Fung B.C.M., Wang K., Ester M. 2003. Hierarchical Document Clustering Using Frequent Itemsets, In Proceedings of the 3rd SIAM International Conference on Data Mining.

[4] (Han, Kamber, 2001) Han J. and Kamber M., 2001, Data Mining: Concepts and Techniques, Academic Press.

[5] (Han et al., 1997) Han E.H, Karypis G., Kumar V., Mobasher B. 1997, Clustering Based on Association Rule Hypergraphs, In Research Issues on Data Mining and Knowledge Discovery.

[6] (Hartigan 1975) Hartigan J. A. 1975, Clustering Algorithms. John Wiley & Sons, New York, NY.

[7] (Jain et al., 1999) Jain A.K., Murty M.N., and Flynn P.J., 1999, Data Clustering: A Review, ACM Computing Surveys, ACM, Vol. 31, No 3, September 1999, pp. 264-323.

[8] (Jing, Ng and Zhexue, 2007) Jing L., Ng M.K., Zhexue J, 2007. An Entropy Weighting k-Means Algorithm for Subspace Clustering of High-Dimensional Sparse Data, IEEE Transactions on Knowledge and Data Engineering, Vol. 19, No. 8, pp. 1026-1041.

[9] (Kanellopoulos et al. 2007) Kanellopoulos Y., Antonellis P., Tjortjis C., Makris C., 2007, k-Attractors, A Clustering Algorithm for Software Measurement Data Analysis, In Proceedings of IEEE 19th International Conference on Tools for Artificial Intelligence (ICTAI 2007), IEEE Computer Society Press.

[10] (Karypis and Kumar, 1998) Karypis G. and Kumar V., 1998, Multilevel k-way partitioning scheme for irregular graphs. Journal of Parallel and Distributed Computing, Vol. 48, No. 1, pp. 96–129.

[11] (Kosters et al., 1999) Kosters W.A., Marchiori E. and. Oerlemans A.A.J. 1999. Mining Clusters with Association Rules, The Third Symposium on in Intelligent Data Analysis (IDA99), pp: 39-50, Lecture Notes in Computer Science 1642, Springer.

[12] (Li et al., 2007) Li W., Ng W.-K, Liu Y. and Ong K.-L., 2007, Enhancing the Effectiveness of Clustering with Spectra Analysis, IEEE Transactions on Knowledge and Data Engineering. Vol. 17, No. 7, pp. 887-902.

[13] (Lin et. al., 2005) Lin C.-R., Liu K.-H., Chen M.-S. 2005. Dual Clustering: Integrating Data Clustering over Optimization and Constraint Domains. IEEE Transactions on Knowledge and Data Engineering. Vol. 17, no.5, pp. 628-637.

[14] (Manilla et. al. 1995) Manilla H., Toivonen H., and Verkamo A.I., 1995, Discovering frequent episodes in sequences, In Proceedings of the First International Conference on Knowledge Discovery and Data Mining, pages 210-215, AAAI Press.

[15] (Patrikainen and Meila 2006) Patrikainen A., Meila M. 2006. Comparing Subspace Clusterings, IEEE Transactions on Knowledge and Data Engineering, Vol.18, no.7, pp. 902-916.

[16] (Wang, Xu, Liu, 1999) Wang K., Xu C., Liu B., 1999, Clustering Transactions Using Large Items, In Proceedings of the 8th ACM International Conference on Information and Knowledge Management, pp.483-490.

[17] (Witten, Frank, 2005) Witten I.H., Frank E. 2005 Data Mining: Practical Machine Learning Tools and Techniques (Second Edition). San Francisco: Morgan Kaufmann.

[18] (Zhuang, Dai, 2004) Zhuang L., Dai H. 2004. A Maximal Frequent Itemset Approach for Web Document Clustering, In Proceedings of the 4th IEEE International Conference on Computer and Information Technology (IEEE CIT'04).

[19] (Orange) Orange, http://www.ailab.si/orange/datasets.asp

[20] (Weka) Weka, http://www.cs.waikato.ac.nz/ml/weka/

[21] (Machine Learning Datasets) Machine Learning Datasets, http://mlearn.ics.uci.edu/