An Architecture and Algorithms for Multi-Run Clustering

Rachsuda Jiamthapthaksin, Christoph F. Eick, and Vadeerat Rinsurongkawong

Abstract—**This paper addresses two main challenges for clustering which require extensive human effort: selecting appropriate parameters for an arbitrary clustering algorithm and identifying alternative clusters. We propose an architecture and a concrete system *MR-CLEVER* for multi-run clustering that integrates active learning with clustering algorithms. The key hypothesis of this work is that better clustering results can be obtained by combining clusters that originate from multiple runs of clustering algorithms. By defining states that represent parameter settings of a clustering algorithm, the proposed architecture actively learns a state utility function. The utility of a parameter setting is assessed based on clustering run-time, quality and novelty of the obtained clusters. Furthermore, the utility function plays an important role in guiding the clustering algorithm to seek novel solutions. Cluster novelty measures are introduced for this purpose. Finally, we also contribute a cluster summarization algorithm that assembles a final clustering as a combination of high-quality clusters originating from multiple runs. Merits of our proposed system are that it is generic and therefore can be used in conjunction with different clustering algorithms, and it reduces human effort for selecting the parameters, for comparing clustering results and for assembling clustering results. We evaluate the proposed system in conjunction with a representative based clustering algorithm namely CLEVER for a challenging data mining task involving an earthquake dataset. The obtained results demonstrate that, in comparison to the best single-run clustering, multi-run clustering discovers solutions of higher quality.**

I. Introduction

Clustering is a very popular descriptive data mining technique that aids describing characteristics of data sets. The goal of clustering is to form groups of objects with similar characteristics. More recently, clustering has been used for scientific discovery, for instance in medical field to identify cancers clusters [1], [2], and in environmental sciences where scientists look for associations between pollutants and other factors [3].

The use of clustering algorithms to aid scientific discovery faces several challenges. Firstly, almost all clustering algorithms require the setting of input parameters which is a non-trivial task and choosing proper values for those parameters is critical for obtaining high-quality clusters. Moreover, many clustering algorithms are probabilistic and different runs, even with the same parameters, lead to different results. For example, many papers have been published to address the problem of choosing proper values for a single input parameter k of the popular K-means algorithm. Furthermore, different data sets have unique implicit characteristics, which require different parameter settings of clustering algorithms. In addition, domain experts frequently look for clusters which exhibit additional characteristics that go far beyond the capabilities of traditional clustering algorithms. A second challenge in employing clustering algorithms is finding alternative clustering; in descriptive data mining, domain scientists are frequently interested in exploring alternative clusters. As for an instance, in [3] a clustering algorithm is applied in finding hotspots in spatial datasets at different granularities, ranging from very local to regional. In general, it not realistic to discover all significant characteristics of a dataset in a single run of a clustering algorithm; even for simple clustering tasks clustering algorithms have to be run multiple times. This establishes the need to analyze the results of several runs of a clustering algorithm which is quite time-consuming. Some recent research focuses on addressing this challenge: alternative clustering [4] constructs a new clustering based on an already known clustering, and ensemble clustering aggregates multiple clusters into a single consolidated clustering [6], [5]. Our work is closely related to the latter approach in that we gather many solutions from multiple runs of a clustering algorithm. However, whereas ensemble approaches assume that all clusters are given beforehand, our approach creates novel and high-quality clusters on the fly: Multi-run clustering seeks for alternative clusters that complement an already given set of clusters.

In this paper, we address the following two challenges in highly automated fashion: 1) selecting parameters of an arbitrary clustering algorithm, and 2) comparing, managing, and summarizing multiple solutions produced by multiple runs of a clustering algorithm. We propose a unified architecture and a concrete system of multi-run clustering that intuitively and automatically select parameters of a clustering algorithm relying on an active learning approach. One of the key components in the proposed architecture is learning the utility of a parameter setting of a clustering algorithm. By representing parameters of a clustering algorithm as states, a state utility function is learnt that guides the search for alternative clustering. The key idea is that our system seeks for novel solutions. When the repeated use of similar parameter settings leads to clusters that were already discovered, the utility of these parameter settings drops, leading to the exploration of alternative parameter

R. Jiamthapthaksin is with University of Houston, TX 77204 USA (phone: 713-743-3350; fax: 713-743-3335; e-mail: rachsuda@cs.uh.edu).

V. Rinsurongkawong is with University of Houston, TX 77204 USA (e-mail: vadeerat@cs.uh.edu).

C. F. Eick is with University of Houston, TX 77204 USA (e-mail: ceick@cs.uh.edu).

settings. The paper proposes several measures and algorithms including:

1. A new measure, called *Novelty* to evaluate a clustering; *Novelty* takes into account two measures: cluster similarity and quality.
2. Cluster List Management algorithm that selectively stores incoming clusters in a cluster repository.
3. Dominance-guided cluster Reduction algorithm, that creates a final clustering by assembling “distinct and good” clusters that originate from multiple runs.
4. A utility update function that updates utilities of parameter settings based on new clustering results.

The proposed work provides 3 advantages: firstly, it automatically selects parameters of a clustering algorithm and incrementally obtains and maintains a set of distinct and good clusters. Secondly, it also generates a final clustering automatically. Finally, multi-run clustering can be used to automatically create “better” inputs for ensemble clustering.

The rest of our paper is organized as follows: we describe architecture and a concrete system of multi-run clustering in section 2 and 3, respectively. Section 4 provides a demonstration and experimental evaluation of our proposed system. Section 5 discusses related work and Section 6 summarizes our findings.

II. An Architecture of Multi-Run Clustering System

A. Goals of Multi-Run Clustering

In this paper, we propose multi-run clustering—a novel approach for clustering. The goal of multi-run clustering is to find a set of distinct and high quality clusters that originate from different runs. Architecture of the multi-run clustering system (*MRCS*) composes of four components: state utility learning (*L*), a clustering algorithm (*CA*), storage unit (*SU*) and cluster summarization unit (*CU*), as depicted in Fig. 1, which performs three essential tasks:

1. Finding good parameter settings for a clustering algorithm

2. Determining which clusters—that have been obtained by running a clustering algorithm—are worth saving as candidate solutions, to be considered for inclusion in the final clustering result

3. Generating a final clustering from candidate clusters

The multi-run clustering generates a final clustering *M’* from a dataset *O* by *learning a utility of parameters of a clustering algorithm in a state space, through active exploration and by incrementally storing the different and high-quality clusters M, which are generated from a group of parameters*. Details of the four components and their corresponding roles are given in the following sections. Table I provides commonly used notations in the paper.

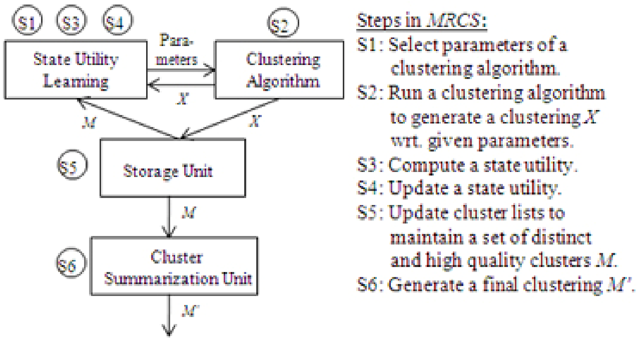

S1
S3
S4
S2
State Utility Learning
Para-meters
Clustering Algorithm
X
M
X
S5
Storage Unit
M
S6
Cluster Summarization Unit
M'
Steps in MRCS:
S1: Select parameters of a clustering algorithm.
S2: Run a clustering algorithm to generate a clustering X wrt. given parameters.
S3: Compute a state utility.
S4: Update a state utility.
S5: Update cluster lists to maintain a set of distinct and high quality clusters M.
S6: Generate a final clustering M'.

Fig. 1. An architecture of multi-run clustering system (*MRCS*)

TABLE I
NOTATION REFERENCE: LIST OF COMMONLY USED NOTATIONS

State utility learning parameters	
$s \in S$	A state in a state space
$\pi \in \Pi$	A policy in a policy space
$U(s)$	A utility of a state
$U'(s)$	A utility update function of a state
α	A learning rate
RCQ	A relative clustering quality function
CLEVER’s parameters	
k	Number of initial clusters
k'	Number of final clusters
p,p'	Sampling size
Multi-run clustering	
$x_i \in X$	A cluster in a clustering X
$y_i \in M$	A cluster in a stored cluster list M
$o_i \in O$	An object in a spatial dataset O
$q(X)$	A fitness value of a clustering
$Reward(x_i)$	A reward of a cluster
$i(x_i)$	An interestingness of a cluster
$P(\pi)$	A probability of applying a policy
M'	A final clustering of *MRCS*
MRS	A set of states of multi-run clustering
L	A state utility learning
CA	A clustering algorithm
SU	A storage unit
CU	A cluster summarization unit

B. State Utility Learning

An essential key of multi-run clustering is to automate parameter selection of an arbitrary clustering algorithm. We employ a state utility learning (*L*) to find good values for the parameters leading to novel and high-quality clusterings. Firstly, we give all definitions regarding of the use of state utility learning in parameter selection and parameter evaluation, and then explain its working principles in *MRCS*.

Definition 1: States in MRCS—A state s in a state space S ($S \in R^{2bm}$) contains ranges of parameters of a clustering algorithm:

$$s = \{s_{1\_min}, s_{1\_max}, \ldots, s_{m\_min}, s_{m\_max}\},\ s_i \in \Re^{2b}$$

where m is the number of parameters of a clustering algorithm and b is the number of bins in a parameter space. Parameter values are discretized into several ranges to reduce the number of states.

Definition 2: Policies Π in MRCS
Let
$\pi \in \{\pi_1, \ldots, \pi_r\} \subseteq \Pi$
π denotes a policy which is defined as follows,
$\pi : MRS \rightarrow S$
where *MRS* denotes a set of states of *MRCS*.

Definition 3: A state utility function U
$U: S \rightarrow \Re$
$U(s)$ denotes a utility of a state s in the state space S.

Definition 4: A relative clustering quality function RCQ
Let
$O = \{o_1, \ldots, o_n\}$ be a dataset.
$X = \{x_1, x_2, \ldots, x_k\}$, $x_i \cap x_j = \emptyset$, $(i \neq j)$, $x_i \subseteq O$, and $\bigcup_{i=1}^{k} x_i \subseteq O$ be a current clustering generated by *CA*.
$M = \{x_1, x_2, \ldots, x_p\}$, $x_i \subseteq O$ be a set of clusters stored in *MRCS*.
We define a relative clustering quality function (*RCQ*) as
$RCQ: X \times M \rightarrow \Re$

In particular, our approach uses *REQ* defined in (1) to compute a utility of the state s. *REQ* evaluates the utility of a state considering three measures: novelty of X with respect to M, computational time ($Speed(X)$) needed and quality of the clustering ($q(X)$).

$$RCQ(M, X) = Novelty(X, M) \times Speed(X) \times q(X) \quad (1)$$

Definition 5: Utility Update U'—the following update rule is used to update state utilities:

$$U'(s) = [(1-\alpha) \cdot U(s)] + (\alpha \cdot RCQ(s)) \quad (2)$$

where α is the learning rate .

By representing parameters of *CA* as states (in Definition 1), *MRCS* uses a policy π (in Definition 2) to select a state s *to be explored next* based on utilities (in Definition 3) of other factors. Then, *MRCS* chooses a particular parameter setting of *CA* at random from the state range of s, and applied it to *CA* in order to produce a clustering X. After that *MRCS* evaluates X using *RCQ* in (1) (in Definition 4); *RCQ* assesses a utility of s based on the quality of X relative to a set of previously found clusters M. Finally, it applies the update rule in (2) (in Definition 5) to update the utility of s based on the relative clustering quality obtained from X in the context of M.

C. Storage Unit and Cluster Summarization Unit

Storage unit (SU) maintains a set of distinct, good candidate clusters in incremental fashion during multiple runs of the clustering algorithm (*CA*). It takes an incoming clustering X produced by *CA* as an input to update its cluster list M.

After multiple runs of parameter selection and clustering, Cluster summarization unit (CU) finally assembles a final clustering; it takes a set of clusters in M as inputs, and generates the final clustering $M' \subset M$.

D. Spatial Clustering Algorithms

In addition to other components in the multi-run clustering we assume that clustering algorithms perform the follow tasks:

Definition 6*: *A clustering algorithm (CA)
Given
$O = \{o_1, \ldots, o_n\}$ as a dataset
A clustering algorithm seeks for a clustering X that maximizes a fitness function $q(X)$.
$X = \{x_1, x_2, \ldots, x_k\}$, $x_i \cap x_j = \emptyset$, $(i \neq j)$, $x_i \subseteq O$, and $\bigcup_{i=1}^{k} x_i \subseteq O$.
Our approach uses reward-based fitness functions in Definition 7 that allow us to compare different clusters (x) and clusterings (X).

Definition 7*: *Fitness Function
$q(X)$ measures the quality of a clustering X as a sum of the rewards obtained from each cluster x_j ($j=1, \ldots, k$), where the interestingness function, $i(x_j)$ computes the interestingness of the cluster x_j, $size(x_j)$ is the number of objects in the cluster x_j, and β is a parameter to tune the significance of cluster size.

$$q(X) = \sum_{j=1}^{k} (Reward(x_i)) \quad (3)$$

$$Reward(x_i) = i(x_j) \times size(x_j)^{\beta} \quad (4)$$

III. The Multi-Run Clustering System MR-CLEVER

In this section, we propose a concrete system of multi-run clustering called *MR-CLEVER* based on the architecture of *MRCS*; the goals of *MR-CLEVER* are to search for distinct clusters automatically, and to enhance cluster results produced by a clustering algorithm. First, we give an overview of CLEVER [9], a clustering algorithm that is a part of *MR-CLEVER*, then we discuss *MR-CLEVER* and its measures and policies in Section B.

A. The CLEVER Clustering Algorithm

CLEVER (CLustEring using representatiVEs and Randomized hill climbing) searches for a clustering that maximizes an externally given, fitness function $q(X)$. The representative-based clustering algorithm forms clusters by assigning objects in a dataset to their closest representatives. The algorithm starts with randomly selecting k' representatives from O where k' denotes the number of initial clusters. It samples p solutions in the neighborhood of the current solution. The neighboring solutions are created using one of the following three operators: insert, delete and replace. Each operator has a certain selection probability and representatives to be manipulated are chosen at random. Next, CLEVER evaluates all the p neighbors, and if there is an improvement, it picks the best among them which becomes the new current solution. To battle premature convergence, CLEVER re-samples $p' > p$ solutions before terminating. It should be mentioned that number of final clusters (k) are not necessarily equal to k'.

CLEVER requires 3 main parameters: the number of initial clusters k' and sampling sizes p and p' which are discretized to form a state space in which states are represented as follows: $\{k'_{min}, k'_{max}, p_{min}, p_{max}, p'_{min}, p'_{max}\}$.

B. MR-CLEVER.

MR-CLEVER is composed of 6 steps (S1 to S6) as depicted in Figure 1. The system iteratively runs t iterations of steps S1 through S5, and then creates the final clustering (step S6).

Pre-processing step. Compute necessary statistics to set up multi-run clustering system. Before we start running multi-run clustering system, background knowledge (statistics) concerning applying the clustering algorithm on a given dataset is required. Therefore, we run m rounds of CLEVER by randomly selecting k', p and p' to compute necessary statistics, such as mean and standard deviation, regarding of clustering quality and runtimes.

Step 1. Select parameters of a clustering algorithm. *MR-CLEVER* uses *policies* given externally by the users to select an anticipated state (CLEVER's parameters) to be explored. The system employs the policies which are chosen probabilistically. After obtaining a state, a parameter setting within ranges of the selected state is selected at random and CLEVER is run for this parameter setting.

An example: Given a set of policies (in Fig. 2) with associated probabilities: $P(\pi 1) = 0.2$, $P(\pi 2) = 0.6$, $P(\pi 3) = 0.2$. We assume that visited states for two iterations of multi-run clustering are $s_1 = \{k'_{min}=1, k'_{max}=10, p_{min}=1, p_{max}=10, p'_{min}=11, p'_{max}=20\}$ and $s_2 = \{k'_{min}=11, k'_{max}=20, p_{min}=41, p_{max}=50, p'_{min}=31, p'_{max}=40\}$ and that s_2 is the state with the highest utility; in the 3rd iteration, *MR-CLEVER* chooses $\pi 2$ at random as a state policy and the policy picks s_2 from the visited states. Next, it further chooses $k'=12$, $p=45$, $q=40$ at random from s_2 as current parameters of CLEVER.

π1. Randomly select a state.
π2. Choose state with the maximum state utility value.
π3. Choose state in the neighborhood of the state having the maximum state utility value.

Fig. 2. Examples of the policies

Step 2. Run CLEVER to generate a clustering with respect to given parameters.

Step 3. Compute a state utility. State utilities are computed using formula 1**.** In particular, *the novelty of a state s in (1) is assessed based on the obtained clustering X, which relies on two measures:*

Novelty measures the degree of novelty of a new clustering X with respect to a set of clusters M stored in the cluster repository (given in (5)). In general, in multi-run clustering, we are interested in finding high-quality clusters that are either different or better than the clusters that have been found so far. Consequently, novelty is assessed using two measures: dissimilarity and enhancement. Both measures compare each cluster in X with the most similar cluster in M. As shown in (6), *Similarity* evaluates the average degree of overlap between clusters in X and the most similar ones in M. On the other hand, *Enhancement* evaluates the quality of X in the context of M as given in (7). In this paper, enhancement assesses number of clusters in X having higher reward per object than their closest competitors in M.

$$Novelty(X,M) = (1\text{–}Similarity(X,M)) \times Enhancement(X,M) \quad (5)$$

where

$$Similarity(X,M) = \frac{\sum_{i=1}^{k} sim(x_i, y_i)}{k} \quad (6)$$

where $sim(x_i, y_i) = \frac{|x_i \cap y_i|}{|x_i \cup y_i|}$, $X = \{x_1,\ldots,x_k\}$, and y_i be the most similar cluster in the stored cluster list M to $x_i \in X$.

$$Enhancement(X,M) = \frac{count\big(rew(x_i) > rew(y_i)\big)}{k} \quad (7)$$

where $rew(c) = \frac{Reward(c_i)}{|c|}$

Speed. The speed function assesses how quickly a clustering algorithm produces a clustering. An interpolation function is applied based on 3 inputs: average and standard deviation of the runtimes and an acceptable runtime, to calculate speed between range of 0 and 1; the speed drops dramatically if runtime is over 70%. First two inputs are computed in the preprocessing step, while last input is given by the users.

```
Let
    $M$ be the current set of multi-run clusters.
    $X$ be a new clustering to be processed for updating $M$.
    $\theta_{sim}$ be a similarity threshold.
    $r_{th}$ be a reward storage threshold.

$X$ will be processed as follows:
FOR $c \in X$ DO
    Let $m$ be the most similar cluster in $M$ to $c$.
IF $sim(m,c) > \theta_{sim}$ AND $Reward(m) < Reward(c)$ THEN
    $replace(m,c,M)$
ELSE IF $Reward(c) > r_{th}$ THEN  $insert(c,M)$
ELSE $discard(c)$;
```

Fig. 3. Cluster List Management algorithm (CLM)

Step 4. Update a state utility. The system updates the utility of the current state by using the utility value computed in S3 in the utility update rule (Definition 5).

Step 5. Update cluster lists to maintain a set of distinct and high quality clusters. Our system uses multiple strategies to incrementally update the cluster list *M*. The experiments in the paper use Cluster List Management algorithm described in Fig. 3. In short, the strategy inserts dissimilar clusters, if their quality is above a given threshold; on the other hand, if a similar cluster exists in *M*, this cluster is only replaced if the cluster in *X* is better than the corresponding cluster in *M*.

```
Let
DEDGE:=$\{(c1,c2)|c1\in M \wedge c2\in M \wedge sim(c1,c2)>\theta_{rem} \wedge$
            $better(c2,c1)\}$
REMCAND:=$\{c|\exists d\ (c,d)\in$DEDGE$\}$
DOMINANT:=$\{c|\exists d\ (d,c)\in$DEDGE $\wedge\ c\notin$REMCAND$\}$
REM:=$\{c|\exists d\ ((c,d)\in$DEDGE $\wedge\ d\in$DOMINANT$)\}$
$Better(c1,c2)\leftrightarrow Reward(c1)>Reward(c2) \vee$
              $(Reward(c1)=Reward(c2) \wedge$
               $clusterNumber(c1)>clusterNumber(c2))$
Remark: Ties have to broken so that DEDGE is always a
DAG; no cycles in DEDGE are allowed to occur.

Input: $M$, $\theta_{rem}$
Output: $M'\subseteq M$

Compute DEDGE from $M$;
Compute REMCAND;
Compute DOMINANT;
WHILE true DO
 {
  Compute REM;
  IF REM=$\varnothing$ THEN EXIT ELSE $M$=$M$/REM;
  Update DEDGE by removing edges of deleted clusters in
  REM;
  Update REMCAND based on DEDGE;
  Update DOMINANT based on DEDGE and REMCAND
}
RETURN($M$).
```

Fig. 4. Dominance-guided Cluster Reduction algorithm (DCR)

Step 6. Generate final clustering. We use Dominance-guided Cluster Reduction algorithm, displayed in Fig. 4, to produce the final clustering *M*' from *M*. After steps 1 to 5 in *MR-CLEVER* are repeatedly run for a while, *M* usually contains a lot of overlapping clusters. The goal of the presented algorithm is to remove clusters from *M* so that *M'* does not contain any pair of clusters (*c*1,*c*2) such that *sim*(*c*1,*c*2)>θ_{rem}, where θ_{rem} is a similarity threshold. The challenge of designing such algorithm is to avoid unnecessary removals of clusters from *M* when generating *M'*. The algorithm relies on *dominant clusters* which are clusters that do not contain any better clusters in their θ_{rem}-neighborhood, and therefore have to remain in *M*. The algorithm loops over the following 2 steps until there is no cluster to be removed in the second step:

1. Compute dominant clusters.
2. Remove clusters in the θ_{rem}-proximity of dominant clusters.

By following this strategy, only those clusters that are similar to dominant clusters are deleted; the other clusters are remained in *M*.

IV. Experimental Evaluation

We illustrate our algorithm on an artificial dataset in section A and evaluate the algorithm in a real world dataset in section B.

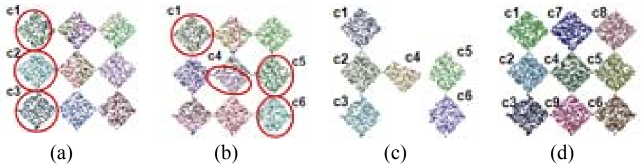
A. Illustration of Multi-Run Clustering on an Artificial Dataset

This experiment illustrates how the proposed multi-run clustering system (*MR-CLEVER*) works. We perform multi-run clustering using CLEVER algorithm with Purity interestingness function [7] on 9Diamonds dataset available in [8]. The dataset consists of 3,000 objects with 9 natural clusters. The Purity function measures a purity degree of each cluster, i.e. a ratio of number of objects in a majority class to the cluster size. The fitness of a clustering is a summation of the purity degrees of all clusters. The parameter settings are shown in Table II; we assume that the number of natural clusters is unknown by the clustering algorithm.

In the 1st run of the multi-run clustering, CLEVER generates 13 clusters (Fig. 5 (a)); clusters *c*1, *c*2, and *c*3 are inserted into *M*. After that CLEVER generates 14 clusters in the 2nd run (Fig. 5 (b)); cluster *c*1 of the 1st run is replaced by *c*1 of this round because of superior purity. Moreover, novel clusters *c*4, *c*5 and *c*6 that also have high purity are inserted in *M*. Fig. 5 (c) depicts distinct and high-purity clusters collected in the cluster repository at the end of the 2nd run. By using insert/replace strategies to manage each incoming clustering (CML in Fig. 3), *MR-CLEVER* can quickly discover the correct 9 clusters (in Fig. 5 (d)) at the end of the 5th run of CLEVER. We observe that CLEVER discovers the correct solution at the 20th run.

TABLE II
Parameter Settings for the 9Diamonds Dataset

β=1.5	θ_{sim}=0.8	θ_{rem}=0.6	Run = 20 rounds
State bound	*k'*=[1,40]	*p*=[1,50]	*p'*=[1,50]
Bin size	*k'*=10	*p*=5	*p'*=5



(a) (b) (c) (d)

Fig. 5. An illustration of *MR-CLEVER*: (a) and (b) depict clustering results of the 1st and 2nd runs respectively, (c) depicts the distinct and high quality clusters in *M* at the end of the 2nd run, and (d) depicts the final clusters produced by *MR-CLEVER*.

B. Evaluation of Multi-Run Clustering on Real Dataset

In this section, we show how *MR-CLEVER* can discover interesting and alternative clusters in spatial data. We employ an earthquake dataset available on the website of the U.S. Geological Survey Earthquake Hazards Program http://earthquake.usgs.gov/. We sampled 4,132 examples of earthquakes dated from January 1986 to November 1991. Information recorded includes the location (longitude, latitude) and the depth (kilometers) of the earthquake.

In this case study, we are interested in discovering places in geographical space where variance of depth of earthquakes is high. In other words, we are interested in areas where deep earthquakes are in close proximity to shallow earthquakes. In this experiment, we use the High Variance function in [10] to find such regions. The interestingness function is defined as follows:

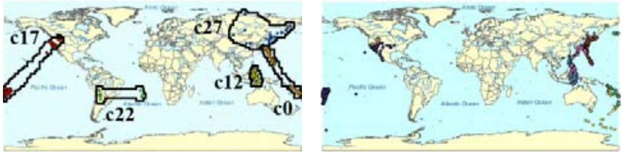
$$i(c) = \left\{ \begin{matrix} 0 & \dfrac{Var(c,z)}{Var(O,z)} \leq th \\ \left(\dfrac{Var(c,z)}{Var(O,z)} - th\right)^{\eta} & otherwise \end{matrix} \right\} \quad (8)$$

where $Var(c,z) = \frac{1}{|c|-1}\sum_{o\epsilon c}(z(o)-\mu_z)^2$

z denotes an attribute of interest in the dataset O which is the depth attribute in the earthquake dataset. th>1 is a setting threshold. 0<η<∞ is a form parameter. μ_z is an average depth of all earthquakes in the cluster c. $Var(c,z)$ denotes a regional variance of z in a cluster c whereas $Var(O,z)$ denotes a global variance of z in the dataset. The parameter settings are shown in Table III.

TABLE III
PARAMETER SETTINGS FOR THE EARTHQUAKE DATASET

β=1.5 th=1.05	η=2.0	θ_{sim}=0.6	r_{th}=3361
Preprocessing run = 5 rounds		Run = 20 rounds	
State bound	k'=[11,80]	p=[1,50]	p'=[1,50]
Bin size	k'=10	p=5	p'=5



(a) (b)

Fig. 6. Multi-run clustering results; (a) depicts top 5 clusters of $X_{TheBestRun}$ (ordered by reward) and (b) depicts clusters in M'.

TABLE IV
INFORMATION OF TOP 5 CLUSTERS OF $X_{THEBESTRUN}$ (ORDERED BY REWARD)

Cluster id	Reward	Interestingness	Size
c17	108524	34	213
c0	15875	19	87
c12	6742	19	50
c22	1807	66	9
c27	1714	13	25

This experiment illustrate 4 benefits of *MR-CLEVER* in: 1) enhancing quality of clusters (by finding similar but better clusters), 2) maintaining high-quality clusters, 3) finding alternative clusters, and 4) finding new high-quality clusters from other runs whereas filtering out low-quality clusters.

Firstly, we analyze behaviors of *MR-CLEVER* by using the best run of CLEVER found at Run No. 2 ($X_{TheBestRun}$). We focus on the top 5 clusters in $X_{TheBestRun}$ as visualized in Fig. 6 (a); the corresponding details: reward, interestingness and size of the clusters are also given in Table IV. According to our observation, we distinguish 3 cases as follows:

Survival Clusters. We find that only one clusters c0, that is inserted into a stored cluster list M, still survives in the final clustering M'.

Non-survival Clusters. We observe that there are two clusters c12 and c17 that are at first inserted in M, but are replaced by similar but higher-reward clusters in the next run.

Discarded Clusters. The last case differs from the 2nd case in that *MR-CLEVER* considers low-reward clusters uninteresting and never inserts them into M; the last two clusters c22 and c27 fall into this case—rewards of the two clusters are lower than r_{th} in Table III.

We further analyze $X_{TheBestRun}$ by comparing them with their corresponding most similar clusters in M' (as shown in Table VI.) Apparently, most clusters in M' are superior to ones in $X_{TheBestRun}$. In particular, *MR-CLEVER* is able to enhance quality of clusters, e.g. finding cluster m5 and m0 which are similar but superior to c12 and c17, respectively. On the other hand *MR-CLEVER* also maintains high quality clusters obtained in the best single run as shown in Item 2; cluster m1 in M' is cluster c0 in $X_{TheBestRun}$. In addition, *MR-CLEVER* weeds out the low-quality clusters as seen in Item 4 and 5; *MR-CLEVER* does not keep any cluster in M' that similar to c22 and c27. Figure 6 (b) also visually confirms absence of clusters c22 and c27 in M'.

TABLE V
INFORMATION OF TOP 10 CLUSTERS IN M' (ORDERED BY REWARD)

Cluster Id	Reward	Interestingness	Cluster Size	Run No.
m0	108865	35	215	0
m30	100569	29	231	13
m1	15875	20	87	2
m15	15740	5	210	6
m40	10979	15	80	19
m33	10570	51	35	15
m9	9570	14	77	5
m5	6777	25	42	1
m39	5846	2	200	17
m2	4691	17	42	0

TABLE VI
COMPARE REWARDS AND SIMILARITIES OF THE TOP 5 CLUSTERS OF $X_{THEBESTRUN}$ AND THE CORRESPONDING MOST SIMILAR CLUSTERS IN M'

Item	The top 5 clusters of $X_{TheBestRun}$		The most similar clusters in M'		Similarity
	Cluster id	Reward	Cluster id	Reward	
1	c17	108524	m0	108865	0.776
2	c0	15875	m1	15875	1
3	c12	6742	m5	6776	0.673
4	c22	1807	N/A	N/A	0
5	c27	1714	m15	15740	0.049

Next, we illustrate an ability of *MR-CLEVER* to discover alternative clusters. Refer to Table V which provides details of the final 10 clusters of *MR-CLEVER* (M'), we consider m0 and m30 alternative clusters. It can be seen from Fig. 7, that these two clusters are highly overlapping (m0 is almost contained in m30.) The two clusters are *alternative* clusters to each other in different aspects; refer to Table V, m30 is superior in terms of cluster size, whereas *m*0 is superior in terms of cluster interestingness and reward. At this stage, the users can decide to keep both or only one of them. It is noted that DCR algorithm in the cluster summarization unit is opened to allow different measures of cluster quality, e.g. using interestingness instead of reward.

Next, we show an ability of *MR-CLEVER* to find new high-quality clusters from other runs. Table VII maps the most similar positive-reward clusters in $X_{TheBestRun}$ to the final clusters in *M'*; with overlapping threshold greater than 30%, there are 70% of the clusters that are found in *M'* but not in the positive-reward clusters of $X_{TheBestRun}$. This implies that *MR-CLEVER* can find 70% of the new and high-quality clusters that do not exist in the best single run. On the other hand, Table VIII shows that with overlapping threshold greater than 20%, there are 42.86% of the clusters that are found in the positive-reward clusters of the best run but not in *M'*. This indicates that *MR-CLEVER* weeds out most of the low quality clusters produced in the best single run.

TABLE VII
THE MOST SIMILAR (POSITIVE REWARD) CLUSTERS IN $X_{THEBESTRUN}$ TO THE FINAL CLUSTERS IN *M'*

Cluster id in *M'*	Cluster id in $X_{TheBestRun}$	*Similarity*
*m*0	*c*17	0.776
*m*1	*c*0	1
*m*2	*c*12	0.15
*m*5	*c*12	0.673
*m*9	*c*6	0.182
*m*15	*c*0	0.286
*m*30	*c*17	0.237
*m*33	*c*0	0.162
*m*39	*c*12	0.244
*m*40	*c*0	0.113

TABLE VIII
THE MOST SIMILAR CLUSTERS IN *M'* TO THE POSITIVE-REWARD CLUSTERS IN $X_{THEBESTRUN}$

Cluster id in $X_{TheBestRun}$	Cluster id in *M'*	*Similarity*
*c*0	*m*1	1
*c*6	*m*9	0.182
*c*12	*m*5	0.673
*c*14	N/A	0
*c*17	*m*0	0.776
*c*22	N/A	0
*c*27	*m*15	0.049

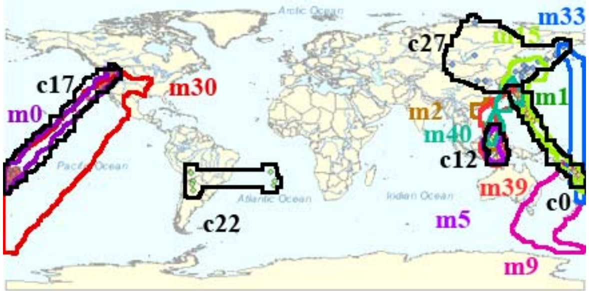


Fig. 7. Overlay the multi-run clustering result (in color) by the top 5 rewards clusters of the best run (in black).

Finally, in Fig. 7, we overlay the multi-run clustering result *M'* (in color) and the top 5 clusters of the best run (in black). The overlap of clusters convince the same argument that *MR-CLEVER* is able to capture the high quality clusters (*c*0, *c*12 and *c*17) from the best run, and to discard some clusters (*c*22 and *c*27) whose reward does not satisfy the threshold. By performing multiple runs, it is able to accumulate novel clusters having high reward that are not found in the best run, e.g. cluster *m*9. This visualization assures us that *CLEVER* cannot find all of the best clusters in a single run, but *MR-CLEVER* can discover them automatically and effectively.

V. RELATED WORK

Active learning has been originally proposed to perform instance selections to enhance performance in classification [11], [12]. Our work adapts active learning to assist parameters selection in clustering. There is some work that uses active learning for cluster enhancement. Klein et al. propose constrained complete-link in [13], a clustering algorithm that actively propagates cannot-link constraints during agglomerative clustering; the merging process results in an implicitly reduction of number of pairwise constraints in the next iteration. Basu et al. present a framework in [14] that actively and explicitly selects a set of representatives by utilizing must-link and cannot-link constraints, and employs them in the partition-based clustering algorithm PCKmeans. Our work is generally similar to the latter work in that it aims to automatically find good parameters. However, the mentioned work focuses on seeking for a set of good initial representatives that produces the optimal clustering, whereas our work centers on finding parameter setting to generate diverse and high-quality clusters.

Reinforcement learning is another approach recently applied to assist clustering ([15], [16]). However, both approaches center on finding the optimal single-run clustering. In addition to the aforementioned work, ReCoM [17] applies reinforcement technique to re-cluster multi-type interrelated data objects. By considering inter-relationship among objects in the different type, it iteratively propagates the clustering result from one type to enhance clustering in the other interrelated type until the clustering converges.

COALA introduced in [4] is an agglomerative clustering algorithm that finds alternative optimal clustering by using prior information of cannot-link instances. Because our approach combines clusters from different runs, it is similar to ensemble clustering and meta-clustering. In this paper, we roughly categorize those approaches into two categories based on aspects of final results produced by the algorithms. The first category assumes that the best clustering is subjective; different users have different opinions in defining the best clustering result. In [18], Caruana et al. early create diverse clusterings, cluster them into groups afterward, and finally let users choose a group of clusterings that is the best for their needs. On the other hand, work belonging to the second category aggregates different clusterings into one consolidated clustering; Gionis et al. propose clustering aggregation algorithms in [6] to generate a final clustering that minimizes the total number of disagreements among all clusterings. Zeng et al. [5] introduce an approach to combine different hard clusterings using probability; objects are assigned to the final clusters based on the probabilities obtained from all the input clusterings. According to the given categories, our work falls into the second category. Whereas the ensemble clustering approach does not address the problem how its input clusterings are generated, our approach finds inputs automatically, and incrementally based on what clusters have been found so far. Another difference is that our approach assumes the presence of an objective function that assesses the clustering quality when creating initial clusters, whereas ensemble approaches use an objective function when creating the final clustering.

VI. Summary and Conclusion

We propose an architecture and a concrete system for multi-run clustering (*MRCS* and *MR-CLEVER*) to cope with parameters selection of a clustering algorithm, and to obtain alternative clusters in highly automated fashion; our approach uses active learning to automate the parameter selection, and various techniques to find both different clusters and good clusters on the fly. At last, a Dominance-guided Cluster Reduction algorithm is proposed, that post-processes clusters from the multiple runs to generate a final clustering by restricting cluster overlap. The merit of the proposed system is that all tasks in multi-run clustering are performed in highly automated fashion: selecting clustering algorithm parameters, running the clustering algorithm, maintaining and analyzing cluster candidates, and assembling final clusters. Moreover, our *MRCS* can be used to produce inputs for ensemble clustering.

The experimental result with the artificial dataset shows that our proposed system performs better than running CLEVER independently multiple times; *MR-CLEVER* identifies the optimal solution more quickly. The experimental result on the real dataset supports our claim that multi-run clustering outperforms single-run clustering with respect to clustering quality. Only 43% of clusters in the best single run occur in the multi-run solution. We also find that 70% of the multi-run clusters do not occur in the best single run clustering. Moreover clusters in the multi-run clustering tend to be significantly better than the corresponding clusters of the best single run clustering. The experimental results also indicate that *MR-CLEVER* can find alternative clusters. In conclusion *MR-CLEVER* can discover additional novel, alternative, high-quality clusters and enhance the quality of clusters found using single-run clusterings.

References

[1] R. T. Ng, J. Sander, and M. C. Sleumer, "Hierarchical cluster analysis of SAGE data for cancer profiling," in *Workshop on Data Mining in Bioinformatics, 7th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, 2001.

[2] E. Malo, R. Salas, M. Catalán, and P. López, "A mixed data clustering algorithm to identify population patterns of cancer mortality in Hijuelas-Chile," in *Lecture Notes in Computer Science, Artificial Intelligence in Medicine*, vol.4594, 2007, pp.190–194.

[3] W. Ding, C. F. Eick, J. Wang, and X. Yuan, "A framework for regional association rule mining in spatial datasets," in *Proc. IEEE Int. Conf. Data Mining*, 2006.

[4] E. Bae, and J. Bailey, "COALA: A novel approach for the extraction of an alternate clustering of high quality and high dissimilarity," in *Proc. 6th Int. Conf. Data Mining*, 2006, pp. 56-62.

[5] Y. Zeng, J. Tang, J. Garcia-Frias, and R. G. Gao, "An adaptive meta-clustering approch: combining the information from different clustering results," in *Proc. IEEE Computer Society Conf. Bioinformatics*, 2002.

[6] A. Gionis, H. Mannila, and P. Tsaparas, "Clustering aggregation," in *Proc. 21st Int. Conf. Data Engineering*, 2005.

[7] C. F. Eick, B. Vaezian, D. Jiang, and J. Wang, "Discovery of interesting regions in spatial datasets using supervised clustering," in *Proc. 10th European Conf. Principles and Practice of Knowledge Discovery in Databases*, 2006.

[8] Data Mining and Machine Learning Group website, University of Houston, Texas. Available: http://www.tlc2.uh.edu/dmmlg/Datasets

[9] C. F. Eick, R. Parmar, W. Ding, T. Stepinki, and J.-P. Nicot, "Finding regional co-location patterns for sets of continuous variables in spatial datasets," in *Proc. 16th ACM SIGSPATIAL Int. Conf. Advances in GIS*, 2008.

[10] V. Rinsurongkawong, and C. F. Eick, "Change analysis in spatial datasets by interestingness comparison," in *ACM SIG SPATIAL Newsletter*, 2008.

[11] D. Angluin, "Queries and concept learning," in *Machine Learning*, vol. 2(4), 1987, pp. 319–342.

[12] L. Atlas, D. A. Cohn, and R. E. Ladner, "Training connectionist networks with queries and selective sampling," in *NIPS*, 1989, pp. 566–573.

[13] D. Klein, S. D. Kamvar, and C. D. Manning, "From instance-level constraints to space-level constraints: making the most of prior knowledge in data clustering," in *Proc. the 9th Int. Conf. Machine Learning*, 2002, pp. 307–314.

[14] S. Basu, A. Banerjee, and R. J. Mooney, "Active semi-supervision for pairwise constrained clustering," in *Proc. 2004 SIAM Int. Conf. Data Mining*, 2004.

[15] A. Bagherjeiran, C. F. Eick, C.-S. Chen, and R. Vilalta, "Adaptive clustering: obtaining better clusters using feedback and past experience," in *Proc. 5th IEEE Int. Conf. Data Mining*, 2005.

[16] C.-H. Oh, E. Ikeda, K. Honda, and H. Ichihashi, "Parameter specification for fuzzy clustering by Q-learning," in *Proc. Int. Joint Conf. Neural Networks*, 2000.

[17] J. Wang, H. Zeng, Z. Chen, H. Lu, L. Tao, and W.-Y. Ma, "ReCoM: reinforcement clustering of multi-type interrelated data objects," in *Proc. 26th Annu. Int. ACM SIGIR Conf. Research and development in information retrieval*, 2003, pp. 274–281.

[18] R. Caruana, M. Elhawary, N. Nguyen, and C. Smith, "Meta clustering," in *Proc. 6th IEEE Int. Conf. Data Mining*, 2006.