2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

An algorithm for scalable clustering: Ensemble Rapid Centroid Estimation

Mitchell Yuwono*, Steven W. Su†, Bruce D. Moulton‡, Ying Guo§, Hung T. Nguyen¶

***Abstract*—This paper describes a new algorithm, called Ensemble Rapid Centroid Estimation (ERCE), designed to handle large-scale non-convex cluster optimization tasks, and estimate the number of clusters with quasi-linear complexity. ERCE stems from a recently developed Rapid Centroid Estimation (RCE) algorithm. RCE was originally developed as a lightweight simplification of the Particle Swarm Clustering (PSC) algorithm. RCE retained the quality of PSC, greatly reduced the computational complexity, and increased the stability. However, RCE has certain limitations with respect to complexity, and is unsuitable for non-convex clusters. The new ERCE algorithm presented here addresses these limitations.**

I. Introduction

CLUSTERING is the unsupervised classification of patterns (observations, data items, or feature vectors) into groups (clusters), based on a measure of similarity. Clustering has proven to be useful in exploratory pattern-analysis, grouping, decision-making, machine learning, data mining, document analysis, image segmentation, and pattern classification [1]. Attempts to develop and improve clustering methods, however, are complicated by the fact that there is no general consensus on how to define/classify natural groupings, where each member of a group is in some way "more similar" to other members of the same group — it has been argued that the notion of "similarity" lacks objectivity [2], [3].

Prior research has shown that stochastic search heuristics, and in particular, particle swarm optimization (PSO), are capable of achieving relatively high levels of clustering quality [4]–[8]. Van Der Merwe reported that PSO could achieve superior results when used to refine the clustering result of k-means on benchmark datasets [4]. Inspired by the success of PSO, Particle Swarm Clustering (PSC) [5] and modified PSC (mPSC) [6] were proposed as PSO variants specifically devised for clustering problems.

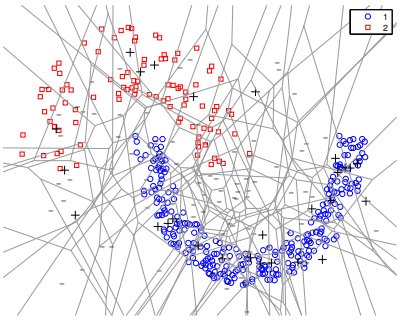
Rapid Centroid Estimation (RCE), an algorithm based on PSC, was proposed in 2013 and reported to increase the efficiency of PSC by providing leaner computational complexity and higher stability [8]. However, we have observed that RCE does not scale well during parallel processing. A further limitation of RCE is that it is suitable only for Gaussian clusters.

*Mitchell Yuwono, †Steven W. Su, ‡Bruce D. Moulton, and ¶Hung. T. Nguyen are with the Faculty of Engineering and Information Technology, University of Technology, Sydney, New South Wales, Australia. §Ying Guo is with the Commonwealth Scientific and Industrial Research Organisation (CSIRO), Marsfield, New South Wales, Australia. (email: *mitchellyuwono@gmail.com; §Ying.Guo@csiro.au; {†Steven.Su, ‡Bruce.Moulton, ¶Hung.Nguyen}@uts.edu.au).

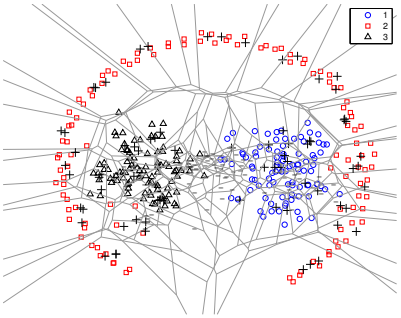
This paper describes research that sought to address the above limitations. An outcome of this research is the newly developed Ensemble RCE (ERCE) algorithm. ERCE differs from RCE in the following ways

1) ERCE further simplifies RCE's update rules and reduces its overall memory-usage and computational complexity,
2) ERCE employs an efficient hybrid ensemble aggregation technique using [9]–[11] which allows it to handle non-convex clusters and estimate the number of clusters in larger datasets.
3) ERCE increases the diversity of particles during swarm mode, by using the concept of "charged particles".

Examples of ERCE's clustering capability are shown in Figure 1.



(a) Half rings dataset. Number of particles = 15, number of swarms = 20, distance metric = euclidean (ℓ_2). Purity = 100%.



(b) Path-based dataset. Number of particles = 50, number of swarms = 5, distance metric = euclidean (ℓ_2). Purity = 97.33%

Fig. 1. Examples of ERCE's clustering capabilities, shown using two different synthetic datasets. During clustering, each ERCE swarm returns locally optimum voronoi tessellations, which are shown here as lines overlayed on the plots. An efficient hybrid ensemble aggregation technique is applied for aggregating these ensemble tessellations. $(+)$ and $(-)$ signs indicate positively and negatively charged particles, respectively.

978-1-4799-1488-3/14/$31.00 ©2014 IEEE

Section II of this paper discusses related prior research, including ensemble methods, PSC, mPSC, and RCE. Section III describes ERCE. Section IV provides a benchmark analysis, which compares ERCE with other methods. Section V focusses on issues relating to computational complexity. Section VI provides conclusions and avenues for further research.

II. Related Prior Research

A. Ensemble Methods

Clustering algorithms can be broadly subdivided into two types: partitional and hierarchical [2]. Partitional clustering defines clusters as *compact partitions*, while hierarchical clustering defines clusters as groups of *connected components* [2]. Commonly applied hierarchical algorithms include the single-linkage (SL), complete-linkage (CL), average-linkage (AL), and ward-linkage (WL) algorithms. Recent studies show that natural clusters can be recovered by employing ensemble methods which combine both the partitional and hierarchical paradigms [3], [9]–[12]. Listed below are four ensemble methods:

1) Evidence Accumulation (EAC)

The EAC was originally proposed by Fred and Jain [12]. The idea is to combine the results of multiple clusterings into a single data partition, by viewing each clustering result as an independent evidence of data organization. The proposed strategy follows a split-and-merge approach.

EAC treats the co-occurrences of pairs of patterns in the same cluster as votes for their association. The partition of n patterns are mapped into a $n \times n$ co-association matrix,

$$\mathcal{C}_{EAC}(i,j) = \frac{n_{ij}}{N}, \tag{1}$$

where n_{ij} denotes the number of times the pattern pair i and j is assigned to the same cluster among the N clustering results.

The natural cluster is recovered by performing agglomerative clustering on the co-association matrix. The final partition is effected by the characteristic of the selected agglomerative algorithm (e.g. SL biases towards connectedness, while AL biases towards compactness). The optimum cut can be identified with the highest lifetime criterion [12].

2) Weighted Evidence Accumulation (WEAC)

Weighted Evidence Accumulation (WEAC) was proposed by Duarte in 2005 to improve the voting mechanism with the inclusion of internal and relative cluster validity indices to weigh multiple clustering results [11]. Given a crisp binary membership matrix from the q^{th} clustering, $\mathsf{U}_q \in [0\,1]$, the co-association matrix is computed as follows,

$$\mathcal{C}_{WEAC} = \frac{\sum_{q=1}^{N} w_q \mathsf{U}_q^T \mathsf{U}_q}{\sum_{q=1}^{N} w_q}, \tag{2}$$

where w_q is a scalar denoting the degree of importance (weight) of the q^{th} clustering result.

3) Fuzzy Evidence Accumulation (fEAC)

Wang proposes the Fuzzy EAC (fEAC) as the extension of EAC for fuzzy clusters [10]. A fuzzy clustering is represented by a fuzzy membership matrix $\mathbf{U}$, where each element u_{tj} defines the membership of the data y_j in the t^{th} cluster. The co-association matrix can be calculated as follows,

$$\mathcal{C}_{fEAC} = \sum_{q=1}^{N} \mathbf{U}_q^T \mathbf{U}_q, \tag{3}$$

where q denotes the clustering solution index. The aggregation product uses the minimum t-norm product [10],

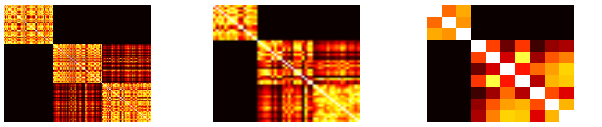
$$u_{ti,q} u_{tj,q} = \sum_{t=1}^{k_q} \min[u_{ti,q}, u_{tj,q}], \tag{4}$$

which is simply the minimum membership of the pattern pair i and j over all cluster indices, $t = \{1, \ldots, k_q\}$. k_q denotes the number of clusters in the q^{th} clustering result.

4) Co-Association Tree (CA-tree)

The CA-tree is proposed by Wang in 2011 [9] in order to reduce the computation and memory complexity of EAC to extend its applicability to larger datasets. The CA-tree applies compression to the co-association matrix in a way that only important representative nodes are retained.

The CA-tree constructs a hierarchical structure similar to a dendrogram using the base cluster label vectors. The CA-tree construction process uses an recursive top-down clustering followed by an efficient bottom-up distance calculation [9]. An illustration showing the effect of the CA-tree compression on a resulting co-association matrix is shown in Figure 2.



(a) $\mathcal{C}_{fEAC}$ (original) (b) $\mathcal{C}_{cfEAC}$ (th=0.15) (c) $\mathcal{C}_{cfEAC}$ (th=0.35)

Fig. 2. An illustration showing the construction of CA-tree and its compression effect on the resulting co-association matrix on different thresholds.

B. PSC and mPSC

1) General Principles

Particle Swarm Clustering (PSC) variants can be viewed as a special modification of PSO, devised specifically for clustering [5]. Conventional PSO clustering assigns each particle to be a representative of a candidate solution [4]. PSC, on the other hand, assigns each particle as a potential centroid candidate [5].

Given a data matrix $\mathbf{Y}$,

$$\mathbf{Y} = \begin{bmatrix} \mathbf{y}_1 & \ldots & \mathbf{y}_j & \ldots & \mathbf{y}_{n_j} \end{bmatrix}, \tag{5}$$

where j denotes the observation index, n_j denotes the number of data (volume); and a particle position matrix $\mathbf{X}$,

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 & \ldots & \mathbf{x}_i & \ldots & \mathbf{x}_{n_i} \end{bmatrix}, \tag{6}$$

where i denotes the particle index, n_i denotes the number of particles; PSC variants performs high dimensional voronoi

tessellation on the data such that each observation in $\mathbf{Y}$ is mapped to the *nearest* particle. In other words, each particle $\mathbf{x}_i$ corresponds to a voronoi cell of the set $\mathbb{C}_i$,

$$\mathbb{C}_{1,\dots,n_i} = \begin{bmatrix}\mathbb{C}_1 & \dots & \mathbb{C}_i & \dots & \mathbb{C}_{n_i}\end{bmatrix}, \emptyset \subseteq \mathbb{C}_{1,\dots,n_i}, \quad (7)$$

which may contain empty sets. The clustered set,

$$\mathbb{C}_{\mathbf{X}} = \mathbb{C}_{r,\dots,n_c} \cap \mathbb{C}_{i,\dots,n_i}, \emptyset \not\subseteq \mathbb{C}_{\mathbf{X}}, \quad (8)$$

is the sets in $\mathbb{C}_{i,\dots,n_i}$ which partitions $\mathbb{Y}$ to n_c non-empty clusters.

The core mechanics of PSC can be explained as follows. Each particle position is recursively updated for each iteration t,

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \Delta\mathbf{x}_i(t+1), \quad (9)$$

$$\Delta\mathbf{x}_i(t+1) = \omega(t)\Delta\mathbf{x}_i(t) + \mathbf{v}_i(t), \quad (10)$$

$$\Delta\mathbf{x}_i(t+1) \in [-\Delta\mathbf{x}_{max}, +\Delta\mathbf{x}_{max}] \quad (11)$$

where $\omega(t)$ represents inertia weight, usually set to a geometrically decreasing function, $\mathbf{x}_i(t)$ and $\Delta\mathbf{x}_i(t)$ denote the position and displacement vectors of a particle i, $\Delta\mathbf{x}$ is lower- and upper-bounded by $\pm\Delta\mathbf{x}_{max}$ which is set to a percentage relative to the search space, and $\mathbf{v}_i(t)$ denotes the direction vector which determines the movement of a particle.

When any of the particles moves, the distance matrix, which measures the pairwise distances between particles and data points, is calculated. This matrix is used to update the *cognitive** matrix, *social*† matrix, and *self-organizing*‡ matrix which is equivalent to the cluster membership.

***Cognitive** ability refers to the ability of each particle to remember the closest position for each data as it moves around the search space. The *cognitive* position vectors are stored in the cognitive matrix $\mathbf{P}_{d\times n_j\times n_i}$.

†**Social** term refers to the ability of each data to remember the historical position of a particle that has been closest to the data during optimization. The *social* position vectors are stored in the social matrix $\mathbf{G}_{d\times n_j}$.

‡**Self-organizing** term refers to the ability of each data to attract the nearest particle. PSC performs voronoi tessellation on the input space using its particles as the voronoi cells. The self-organizing matrix $\mathsf{U}_{n_i\times n_j}$ is equivalent to the k-means (or Hard C-Means, HCM) membership matrix.

At $t = 0$, $\mathbf{X}$ is initialized at random inside the search space, while $\Delta\mathbf{X}$ is set to a zero matrix. The matrices $\mathbf{P}$, $\mathbf{G}$, and U are calculated based on the initial $\mathbf{X}$. Afterwards the aforementioned routines are iteratively computed on each iteration until maximum iteration or equilibrium [5] is reached.

The pseudocode can be written in Algorithm 1.

2) Particle Swarm Clustering (PSC)

PSC particle direction vector is computed as follows. In each iteration the *cognitive* $\mathbf{co}_i^j(t)$, *social* $\mathbf{sc}_i^j(t)$, and *self-organizing* $\mathbf{so}_i^j(t)$ terms are computed for a data vector j,

$$\mathbf{co}_i^j(t) = \mathsf{u}_{i,j}K_1\varphi_1(\mathbf{p}_i^j(t) - \mathbf{x}_i(t)), \quad (12)$$

$$\mathbf{sc}_i^j(t) = \mathsf{u}_{i,j}K_2\varphi_2\left(\mathbf{g}_j(t) - \mathbf{x}_i(t)\right), \quad (13)$$

$$\mathbf{so}_i^j(t) = \mathsf{u}_{i,j}K_3\varphi_3\left(\mathbf{y}_j - \mathbf{x}_i(t)\right). \quad (14)$$

Algorithm 1 PSC variants pseudocode

```
Initialize X, ΔX, P, G, and U.
while t <maximum iteration or stopping criterion not met
  for i = 1, i < n_i, i++
    for j = 1, j < n_j, j++
      Update Δx(t) and x(t).
      Calculate distance matrix,
      Update P, G, and U.
    end for
  end for
  t++;
end while
return U.
```

The direction vector of each PSC particle with more than one member is simply a sum of all the term weighted by scalar uniform random numbers,

$$\mathbf{v}_{\mathbf{psc}_i^j}(t) = \begin{cases}\mathbf{co}_i^j(t) + \mathbf{sc}_i^j(t) + \mathbf{so}_i^j(t) & \text{if } \mathbb{C}_i \notin \emptyset \\ \mathbf{x}_{Iwin}(t) - \mathbf{x}_i(t) & \text{if } \mathbb{C}_i \in \emptyset\end{cases}, \quad (15)$$

where $K_{1,\dots,3}$ are user-set parameters. Guidelines for selecting the appropriate values for $K_{1,\dots,3}$ are noted in [5]. Any particle i with zero members is redirected towards the winning particle I_{win}, which represents the cluster with the most members.

3) Modified Particle Swarm Clustering (mPSC)

Szabo proposed a modification to the original PSC called the Modified PSC (mPSC) in 2010 [6]. mPSC sets the inertia weight $\forall t, \omega(t) = 0$ and displacement vector limits to $\Delta\mathbf{X} \in [-\infty, +\infty]$. The calculation of the direction vector and flow of the algorithm follows that of PSC, as described in Section II-B2 and Algorithm 1.

C. Rapid Centroid Estimation (RCE)

RCE is broadly based on the PSC algorithm but is reconfigured to require less computational complexity, without sacrificing its optimization capability [8]. The principles of RCE are summarized as follows:

1) Particle positions are updated only once per iteration.
2) The distance matrix and best positions are updated after all particle positions are updated.
3) RCE introduces the §**minimum** term which stores the best particles that minimize a user defined objective function. The objective function is generally defined as, but not restricted to, the average intra-cluster distances which is implemented as follows,

$$AWD(\mathbb{C}) = \frac{1}{n_j}\sum_{j=1}^{n_j}\sum_{i=1}^{n_i}\mathsf{u}_{ij}d(\mathbf{y}_j, \mathbf{x}_i), \quad (16)$$

which measures the sum of all distances between each data j to the nearest centroid normalized by the number of data.

The minimum matrix returned by RCE is simply,

$$\forall t, \mathbf{M} = \arg\min_{\mathbf{X}} AWD(\forall\mathbb{C}_{\mathbf{X}}(t) \notin \emptyset), \quad (17)$$

which is a set of all non-empty particles in $\mathbf{X}$ that minimizes the average intra-cluster distances over all

iterations. The optimum self-organizing matrix is referred to in this paper as $\mathsf{U}^{\mathbf{M}}$.

RCE particle direction vector is computed in the following steps. In each iteration the *cognitive** $\mathbf{co}_i(t)$, *social*† $\mathbf{sc}_i(t)$, *self-organizing*‡ $\mathbf{so}_i(t)$, and *minimum*§ $\mathbf{mi}_i(t)$ terms are computed for all data vector j for each particles with at least one member,

$$\mathbf{co}_i(t) = \frac{\sum_{j=1}^{n_j} \mathsf{u}_{i,j}(\mathbf{p}_i^j(t) - \mathbf{x}_i(t))}{\sum_{j=1}^{n_j} \mathsf{u}_{i,j}}, \quad (18)$$

$$\mathbf{sc}_i(t) = \frac{\sum_{j=1}^{n_j} \mathsf{u}_{i,j}(\mathbf{g}_j(t) - \mathbf{x}_i(t))}{\sum_{j=1}^{n_j} \mathsf{u}_{i,j}}, \quad (19)$$

$$\mathbf{so}_i(t) = \frac{\sum_{j=1}^{n_j} \mathsf{u}_{i,j}(\mathbf{y}_j - \mathbf{x}_i(t))}{\sum_{j=1}^{n_j} \mathsf{u}_{i,j}}, \quad (20)$$

$$\mathbf{mi}_i(t) = \frac{\sum_{k=1}^{n_k} \mathsf{u}_{i,k}^{\mathbf{M}}(\mathbf{m}_k - \mathbf{x}_i(t))}{\sum_{k=1}^{n_k} \mathsf{u}_{i,k}}, \quad (21)$$

where n_k denotes the number of vectors in the $\mathbf{M}$ matrix, and $\mathsf{u}_{i,k}^{\mathbf{M}}$ denotes the binary membership value of the i^{th} particle to the k^{th} vector in the $\mathbf{M}$ matrix.

The direction vector of each RCE particle is simply a sum of all terms weighted by a uniform random vector $\mathbf{\Phi}$,

$$\mathbf{v}_{\mathbf{rce}i}(t) = \begin{cases} \mathbf{\Phi} \circ (\mathbf{co}_i(t) + \mathbf{sc}_i(t) \\ \quad + \mathbf{so}_i(t) + \mathbf{mi}_i(t)) & \text{if } \mathbb{C}_i \notin \emptyset\,, \\ \mathbf{x}_{Iwin}(t) - \mathbf{x}_i(t) & \text{if } \mathbb{C}_i \in \emptyset \end{cases} \quad (22)$$

where the $\circ$ operator denotes Hadamard product.

In addition, the *substitution* and *swarm* strategies are introduced to further improve RCE's search reliability [8].

1) substitution strategy

The purpose of the *substitution* strategy is to force particles in a search space to reach alternate equilibrium positions by introducing position instability. For each particle i,

$$\begin{aligned} \mathbf{x}_i(t+1) &= \begin{cases} \mathbf{x}_{Iwin}(t+1) + \mathbf{N}(0,\sigma) & \text{if } \varphi < \varepsilon \\ \mathbf{x}_i(t+1) & \text{otherwise} \end{cases}, \\ \Delta\mathbf{x}_i(t+1) &= \begin{cases} 0 & \text{if } \varphi < \varepsilon \\ \Delta\mathbf{x}_i(t+1) & \text{otherwise} \end{cases}, \end{aligned} \quad (23)$$

is calculated after its position is updated. In eq. (23) φ denotes a uniform random number $\varphi \in \{0, 1\}$, $\mathbf{x}_{Iwin}$ is the position of the winning particle, and $\mathbf{N}(0, \sigma)$ is a Gaussian random vector with mean 0, and σ equal to the empirical standard deviation of the search space, and ε denotes the *substitution* probability parameter. Larger ε increases the *substitution* frequency. Optimal ε values lie between $0.01 \leq \varepsilon \leq 0.05$ [8].

2) swarm strategy

The swarm strategy is simply a collaborative parallel search between RCE groups. In the swarm mode each RCE group shares its minimum matrix such that,

$$\mathbf{M}^{swarm} = \bigcup_{m=1}^{n_m} \mathbf{M}_m(t), \quad (24)$$

which is the union of all $\mathbf{M}_m$ matrices, $m = \{1, \ldots n_m\}$.

The *minimum* term $\mathbf{mi}$ in the swarm mode is termed the *collective minimum* $\mathbf{cm}$, as follows

$$\mathbf{cm}_{i,m}(t) = \frac{\sum_{k=1}^{n_k} \mathsf{u}_{i,m,k}^{\mathbf{M}^{\mathbf{swarm}}}(\mathbf{m}_k^{swarm} - \mathbf{x}_{i,m}(t))}{\sum_{k=1}^{n_k} \mathsf{u}_{i,m,k}}, \quad (25)$$

where m denotes the index for the m^{th} swarm, $\mathsf{u}_{i,m,k}^{\mathbf{M}^{\mathbf{swarm}}}$ denotes a crisp membership of the i^{th} particle in the m^{th} swarm to the k^{th} vector of the collective minimum matrix.

Accordingly, the direction vector of each RCE particle in the swarm mode is similar to eq. (22) except that $\mathbf{M}^{swarm}$ is used in place of the minimum matrix $\mathbf{M}$.

RCE pseudocode with the substitution and swarm strategies can be written in Algorithm 2.

Algorithm 2 RCE pseudocode

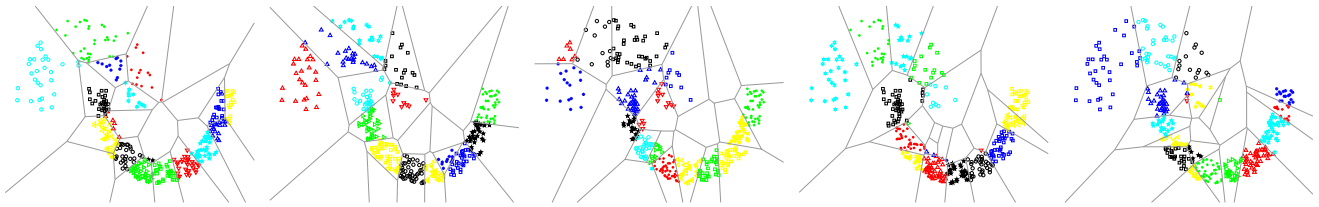
```
Initialize X, ΔX, P, G, Uand M.
while t <maximum iteration or stopping criterion not met
   for m = 1, m < n_m, m++
      for i = 1, i < n_i, i++
         Update Δx(t) and x(t).
         Apply substitution at a given ε value.
      end for
      Calculate distance matrix,
      Update P_m, G, U_m, M_m, and M^swarm.
   end for
   t++;
end while
return U_{M_1,...,nm}.
```

III. Ensemble Rapid Centroid Estimation (ERCE)

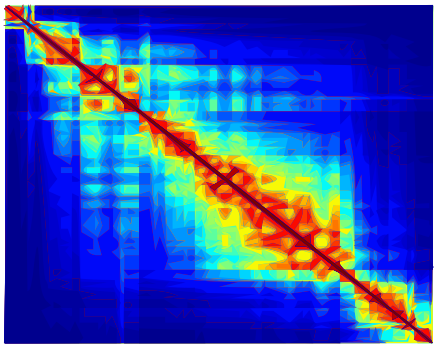
This section explains in detail our proposed ensemble clustering method. The Ensemble Centroid Estimation (ERCE) is proposed as an extension and simplification of RCE when the *swarm* mode is applied. During the clustering stage, ERCE is specially developed to minimize memory consumption and maximize computation efficiency. For the ensemble aggregation, ERCE utilizes a hybrid algorithm using the CA-tree [9], WEAC [11], fEAC [10]. This efficient split-and-merge method allows ERCE to detect non-convex clusters in a quasi-linear complexity. The illustration on the algorithmic flow of the algorithm is shown at Figure 3. The following subsections will discuss how this concept is applied in more detail.

A. Simplification

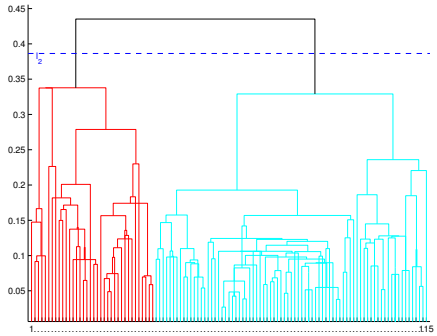
The main challenge for the implementation of RCE as an ensemble algorithm is its memory consumption. The $\mathbf{P}$ and $\mathbf{G}$ matrices in PSC variants, including mPSC, RCE, and swarm RCE contribute largely to the memory complexity during optimization. This is because the size of $\mathbf{P}$ grows linearly as more particles/swarms are added. Taking this into consideration, both the original *cognitive** and *social*† terms are not used in ERCE. The calculations for *cognitive** and *social*† terms are discarded which consequently leads to ERCE having an even less computational complexity to that of RCE.



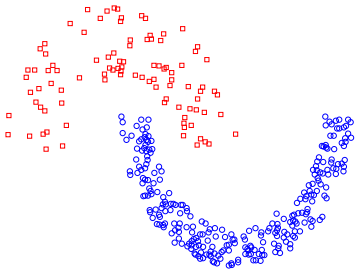
(a) Various voronoi tessellations returned by ERCE.



(b) $\mathcal{C}_{cfWEAC}$ obtained using the proposed CA-tree – fuzzy WEAC Hybrid. The size of the co-association matrix is reduced from 373×373 to 115×115.



(c) The dendrogram (single link) obtained using the proposed method ($\mathcal{C}_{cfWEAC}$ SL). The compressed solution, $\mathsf{U}_{censemble}$, is obtained using the highest lifetime criterion.



(d) The desired decompressed solution, $\mathsf{U}_{ensemble}$, obtained by performing inverse mapping on the CA-tree.

Fig. 3. The algorithmic flow of the proposed method explained using the half rings dataset.

B. Swarm Diversification: The concept of "Charges"

Heyden proposes that the presence of redundant partitions in ensemble clustering aggregation may produce an undesireable bias to the final solution [13]. In order to diversify the particles, ERCE introduces a new concept of *charge* which is inspired by the physical behaviour of electric charges. The diversification strategy, together with the substitution strategy, is devised to create a constant chaotic turbulence in the search space, such that the possibility of creating a duplicate partition is minimized.

There are two types of electric charges – positive and negative. Charges of the opposite polarity will attract one another while charges of the same polarity will repel otherwise. Applying this concept, the ERCE particles can carry either *positive*[¶] or *negative*[‖] charge. The initialization is done at random and each particle remains the same charge until the end of the optimization. Every data is assumed to be negatively charged. The definitions of *positive* and *negative* particles are as follow.

[¶]**Positive** $(+)$ particles are attracted to their member data such that the *self-organizing*[‡] vector is positive,

$$\mathbf{so}_i^+(t) = +\mathbf{so}_i(t). \tag{26}$$

Readers are to refer to (20) for $\mathbf{so}_i(t)$.

[‖]**Negative** $(-)$ particles are repelled by their member data such that the *self-organizing*[‡] movement vector is negative,

$$\mathbf{so}_i^-(t) = -\mathbf{so}_i(t), \tag{27}$$

Readers are to refer to (20) for $\mathbf{so}_i(t)$.

Negative particles are attracted to their nearby non-empty positive particles. For negative particles, the *social*[†] term is redefined as follows,

$$\forall \mathbb{C}_l \notin \emptyset, \mathbf{sc}_i^-(t) = \frac{\sum_{l=1}^{n_l} \mathsf{u}_{i,l} \left(\mathbf{x}_l^+(t) - \mathbf{x}_i^-(t)\right)}{\sum_{l=1}^{n_l} \mathsf{u}_{i,l}}, \tag{28}$$

where l denotes the index of positively charged particles, $\mathsf{u}_{i,l} \in [0,1]$ denotes a crisp membership value of the i^{th} negative particle to the l^{th} nearest positive particle.

The direction vector of each ERCE particle is calculated as follows,

$$\mathbf{v}_{\mathbf{erce}i,m}(t) = \begin{cases} \mathbf{\Phi} \circ \Big(\mathbf{so}_{i,m}^+(t) & \\ +\mathbf{cm}_{i,m}(t)\Big) & \text{if } \mathbb{C}_{i,m} \notin \emptyset, \\ & \text{and } \mathbf{x}_{i,m} \in (+) \\ \mathbf{\Phi} \circ \Big(\mathbf{so}_{i,m}^-(t) & \\ +\mathbf{sc}_{i,m}^-(t) & \\ +\mathbf{cm}_{i,m}(t)\Big) & \text{if } \mathbb{C}_{i,m} \notin \emptyset, \\ & \text{and } \mathbf{x}_{i,m} \in (-) \\ \mathbf{x}_{Iwin,m}(t) - \mathbf{x}_{i,m}(t) & \text{if } \mathbb{C}_{i,m} \in \emptyset \end{cases}, \tag{29}$$

where m denotes the index for the m^{th} swarm, $\mathbf{x}_i \in (+)$ indicates that the particle i is positively charged, $\mathbf{x}_i \in (-)$ indicates that the particle i is negatively charged.

C. Fuzzification of the Distance Matrix

Given a distance matrix between particles and data, $\mathbf{D} = \mathbf{D}(\mathbf{X}, \mathbf{Y})$, the fuzzy membership value for the j^{th} observation with respect to the i^{th} cluster, u_{ij} can be obtained as follows,

$$u_{ij} = \frac{e^{-d_{ij}/(2\lambda_i)}}{\sum_{i=1}^{n_i} e^{-d_{ij}/2\lambda_i}}, \tag{30}$$

where d_{ij} is the distance between the i^{th} particle to the j^{th} observation, and λ_i denotes the bandwidth of the i^{th} cluster center.

Given the fuzzified dissimilarity for the j^{th} data relative to the i^{th} center,

$$\mathcal{D}_{ij} = u_{ij} d_{ij}, \tag{31}$$

and its Shannon entropy,

$$H(u_{ij}) = -u_{ij} \log u_{ij}, \tag{32}$$

where u_{ij} is the fuzzy cluster membership vector of the j^{th} data relative to the i^{th} cell, the optimum bandwidth for each cell $\lambda_{1,\ldots,n_c}$ can be calculated using a compromise between eq. (31) and (32). In other words, for each cluster, $\mathbb{C}_i$, the optimum λ_i can be found by solving a convex optimization problem,

$$\min_{\substack{s.t. \\ \forall i, \lambda_i > 0}} ||H - \mathcal{D}||^2, \tag{33}$$

which optimizes λ_i for all cells, $i = 1, \ldots, n_i$, that best describe the Gaussian probability distribution of the data governed in each corresponding voronoi cell. However, as minimizing eq. 33 requires a relatively high computational complexity, for the sake of efficiency it is usually assumed that every voronoi cell has equal bandwidths, such that $\forall i, \lambda_i = \lambda$.

D. Fuzzy Ensemble Aggregation

The fuzzy ensemble aggregation method can be explained in five steps summarized as follow:

1) Use CA-tree to find the representative nodes from the label vectors.
2) Use the average of the corresponding fuzzy membership representation for each representative nodes.
3) Calculate the weights using average simplified silhouette width criterion (SSWC) [14] and Generalized Dunn Index (GDI) [15],
$$w_q = SSWC_q(\mathbb{C}^{(q)}) \times GDI_q(\mathbb{C}^{(q)}), \tag{34}$$
where $\mathbb{C}^{(q)}$ is the crisp partition obtained from the q^{th} clustering.
4) Calculate the weighted co-association matrix of the representatives nodes,
$$\mathcal{C}_{cfWEAC} = \frac{\sum_{q=1}^{N} w_q \mathcal{U}_q^T \mathcal{U}_q}{\sum_{q=1}^{N} w_q}, \tag{35}$$
where $\mathcal{U}_q$ is the compressed fuzzy membership matrix of the q^{th} clustering, the calculation of $\mathcal{U}_q^T \mathcal{U}_q$ follows eq. (4).
5) Recover the ensemble label matrix $\mathsf{U}_{ensemble}$ from $\mathcal{C}_{cfWEAC}$.

The pseudocode for ERCE is shown in Algorithm 3.

Algorithm 3 ERCE pseudocode

```
%% Ensemble clustering stage (split)
Initialize X, ΔX, U and M.
while t <maximum iteration or stopping criterion not met
   for m = 1, m < n_m, m++
      for i = 1, i < n_i, i++
         Update Δx(t) and x(t).
         Apply substitution at a given ε value.
      end for
      Calculate distance matrix,
      Update U_m, M_m, and M^swarm.
   end for
   t++;
end while
%% Ensemble aggregation stage (merge)
Optimize the fuzzy membership matrices U_{M_{1,...,n_m}}.
Calculate C_cfWEAC.
Recover U_ensemble.
return U_ensemble
```

IV. Benchmark Analysis

The proposed algorithm is evaluated using two real-world datasets from the UCI Machine Learning Repository [16] and a high resolution image clustering problem. The performance of the proposed algorithm will be compared with existing the partitional clustering algorithms including k-means (Hard C-Means - HCM), Fuzzy C-Means (FCM), PSC, mPSC, and swarm RCE. Fixed number of clusters will be supplied for these algorithms.

A comparison to conventional ensemble methods including ensemble HCM EAC, ensemble HCM WEAC, and ensemble FCM fEAC will also be presented. Ensemble HCM WEAC uses SSWC [14] and GDI [15] as the weighting factor. Exponent for the partition matrix in FCM is set to 2. Number of clusterings/repetitions (q) is set to 80 times. the number of clusters for each repetition varies uniformly between 2 to $\sqrt{|N|}$, where $|N|$ is the volume of the dataset. For all ensemble algorithms, the number of clusters are assumed to be unknown.

Parameters for ERCE is set as follow. CA-tree compression threshold is set to 0.1. Number of particles is set to 20, number of swarms is set to 20. Substitution probability, ϵ, is set to 0.03, maximum iteration is set to 20. Positive $(+)$ and negative $(-)$ particles are uniformly spread for each swarm. The number of clusters are assumed to be unknown.

A. Fisher-Iris dataset

The Fisher-Iris dataset [16] contains 150 instances of iris flowers collected in Hawaii. The dataset consists of 50 samples from each of three species of Iris (Iris setosa, Iris virginica and Iris versicolor). Four features were measured. The performance of each algorithms will be evaluated according to the purity and entropy of the clustering results against the ground truth.

Pearson's correlation and WL is selected as the distance metric and ensemble aggregation scheme due to the shapes of the clusters being three elongated Gaussian mixtures. A monte-carlo simulation using 200 trials of various algorithms

is shown in Table I. Shaded cells in Table I indicate ensemble algorithms, dark gray cells indicate ERCE results.

TABLE I
EXPERIMENTAL SUMMARY ON THE FISHER-IRIS DATASET [16].

Algorithm	Purity		Entropy		Time	
	Mean	Std	Mean	Std	Mean	Std
HCM	78.60%	16.80%	0.16	0.03	**1.50E-03**	**6.00E-04**
FCM	89.33%	0.00%	0.27	0.00	8.91E-02	1.84E-01
PSC	79.90%	11.60%	0.31	0.11	3.90E-02	3.00E-04
mPSC	88.60%	10.70%	0.32	0.09	3.80E-02	3.00E-04
RCE	94.61%	4.90%	0.16	0.17	1.60E-03	2.00E-05
Swarm RCE	95.80%	0.66%	0.15	0.02	9.50E-03	6.40E-05
HCM EAC	76.22%	16.70%	0.24	0.05	5.76E-01	2.21E-02
HCM WEAC	82.44%	9.19%	0.22	0.03	1.27E+00	3.86E-02
FCM fEAC	89.02%	2.23%	0.21	0.02	1.54E+00	1.96E-01
ERCE	**96.30%**	**0.51%**	**0.12**	**0.01**	9.69E-01	4.38E-02

From Table I, it can be seen that the ERCE yields best purity relative to the other ensemble algorithms and partitional algorithms. The HCM EAC performs slightly faster on this dataset than ERCE, however it produces the least stable results, which can be seen from its high standard deviation.

B. Optical digits dataset

This dataset is used to benchmark the capability the proposed algorithms discover patterns in data. A good algorithm is expected to find correct assignments based on the data alone. The optical digits dataset [16] describes an optical digit recognition problem from pictorial bitmaps. The data were obtained from a total of 43 people. 32×32 bitmaps are divided into nonoverlapping blocks of 4×4 and the number of on pixels are counted in each block. This generates an input matrix of 8×8 where each element is an integer in the range from 0 to 16. The total volume of the dataset is 5620. The results from HCM EAC(AL) and ERCE is shown in Figure 4.

(a) HCM EAC (AL), 80 repetitions k = 2–75. The algorithm failed to group 9 in an individual cluster. Two clusters of 1 are discovered. Time for optimization = 192.6 seconds.

(b) HCM WEAC (AL), 80 repetitions k = 2–75. Similarly to HCM EAC, HCM WEAC algorithm failed to group 9 in an individual cluster. Time for optimization = 186.1 seconds.

(c) FCM fEAC (AL), 80 repetitions k = 2–75. The algorithm consistently finds only 3 clusters at most after numerous trials. Time for optimization = 351.6 seconds.

(d) ERCE (AL), 15 particles, 20 swarms, 100 iterations. All 10 patterns are discovered and correctly clustered. Time for optimization = 32.66 seconds.

Fig. 4. Clustering results for the Optical Digits recognition dataset

C. Natural image segmentation

One of the important criteria for a clustering algorithm is its capability to handle high volume data. To benchmark the performance of ERCE we use a 634×505 RGB image (volume = 320170, dimension = 3) showing an animal panda and a grass background [17]. Because the colors of the panda (black/white) are contrast to those of its background (green) segmenting this image should be relatively easy. The clustering result of ERCE(SL) is shown in Figure 5.

Fig. 5. ERCE (SL) clustering result for clustering a panda image in the RGB color space (634×505×3). ERCE settings: 15 particles, 20 swarms, 50 iterations. 5 clusters were discovered: the panda's fur (white$^{(1)}$ and black$^{(2)}$); shadows on the panda's fur$^{(3)}$; green background$^{(4)}$; and edges$^{(5)}$. Time for optimization = 56.74 seconds.

The total memory requirement for ERCE for processing this image is 165 Megabytes (MB). Results for other ensemble algorithms are not available as the memory requirements for all methods exceed 5 Terabytes (TB).

V. COMPLEXITY ANALYSIS

A. Memory Complexity

To test the memory consumption of various clustering algorithm on a random data (byte precision, volume = 1 Byte – 1 Megabytes, dimension = 2) can be seen in Figure 6.

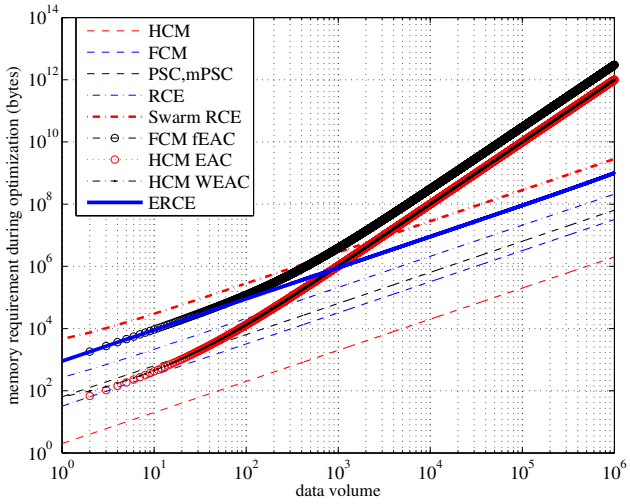


Fig. 6. Memory complexity for clustering 2-dimensional random noise (vol = 1 Byte – 1 Megabytes, d = 2) using various algorithms. The global settings for all algorithms are as follow: the number of representatives (k, n_i) = 30; the number of trials/swarms (n_m) = 30.)

It can be seen from Figure 6 that the lowest memory requirement for non-ensemble methods is achieved by Hard C-Means (HCM, or k-means), followed by Fuzzy C-Means (FCM), PSC, mPSC, RCE, and Swarm RCE. For ensemble methods, ERCE achieves the lowest memory complexity, followed by HCM EAC, HCM WEAC, and FCM fEAC. The scalability problem of EAC-based methods are clearly shown in the graph.

B. Computational Complexity

A known issue with traditional ensemble algorithms — such as the HCM EAC, HCM WEAC, and FCM fEAC, — is that they do not scale well to larger datasets. The main bottleneck for these algorithm is the EAC algorithm [9].

Realizing this issue, the ERCE make use of the EAC algorithm efficiently on the representative nodes which are extracted using CA-tree. The computational complexity of ERCE will be explained in the following paragraphs.

During the clustering stage, ERCE computational complexity is $\mathcal{O}(n_m n_i N)$ for a single iteration, where n_m denotes the number of swarms, n_i denotes the number of particles, and N denotes the number of data. When compared with that of HCM/k-means, it is simply the HCM complexity multiplied by the number of swarms n_m. The complexity for distance fuzzification is $\mathcal{O}(n_m n_i N L_m)$, where L_m is the number function evaluations required for the m^{th} swarm.

During the ensemble aggregation scheme, the ERCE complexity relies on the complexity of both CA-tree ($\mathcal{O}(N)$ [9]) and fcWEAC, $\mathcal{O}(n_i n_m N_{th}^2)$, where n_m denotes the number of swarms, n_i denotes the number of particles, and N_{th} denotes the number of representative nodes at a given threshold, $N_{th} < N$. For higher threshold the volume of the node representatives will be much smaller than the dataset, $N_{th} << N$, which in turn makes the fcWEAC assume a quasi-linear complexity, $\mathcal{O}(n_m n_i log^2(N))$.

The overall complexity of ERCE is therefore,

$$\mathcal{O}(\overbrace{[n_m n_i N(t_{max}+L_m)]}^{\text{clustering + fuzzification}} + \overbrace{[N]}^{\text{CA-tree}} + \overbrace{[n_m n_i log^2(N)]}^{\text{fcWEAC}}), \quad (36)$$

where t_{max} denotes the predefined number of iterations.

VI. Conclusion

This paper proposes the Ensemble Rapid Centroid Estimation (ERCE) as an extension of Rapid Centroid Estimation (RCE) for dealing with large scale ensemble cluster optimization problem. Its formulation leads to the reduction of overall memory and computational complexity, increase in the swarm diversity using the concept of "charged particles" and the ability to handle non-convex clusters and estimate natural grouping for larger dataset in quasi-linear time. The ERCE's low memory and computational complexity are mainly attributed to the modifications on the RCE update rules and the employment of CA-tree [9]. Preliminary experimental result on benchmark datasets including Fisher-Iris, optical digits, and natural image clustering have shown promising results. When clustering a relatively high resolution RGB image, ERCE correctly returns the desired clustering result in less than 1 minute with only 165 Megabytes memory consumption, as opposed to 5 Terabytes using other ensemble algorithms.

Notwithstanding the encouraging results, further work for investigating the stability, reliability and scalability of the ERCE on larger and more complex datasets will be required.

References

[1] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: A review," *ACM Comput. Surv.*, vol. 31, no. 3, pp. 264–323, Sep. 1999. [Online]. Available: http://doi.acm.org/10.1145/331499.331504

[2] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern Recogn. Lett.*, vol. 31, no. 8, pp. 651–666, Jun. 2010. [Online]. Available: http://dx.doi.org/10.1016/j.patrec.2009.09.011

[3] J. Handl and J. Knowles, "An evolutionary approach to multiobjective clustering," *Evolutionary Computation, IEEE Transactions on*, vol. 11, no. 1, pp. 56–76, 2007.

[4] D. W. van der Merwe and A. P. Engelbrecht, "Data clustering using particle swarm optimization," in *Proc. of the 2003 IEEE Congress on Evolutionary Computation, 2003*, vol. 1, Dec. 8–12 2003, pp. 215–220.

[5] S. C. M. Cohen and L. N. de Castro, "Data clustering with particle swarms," in *Proc. of the 2006 IEEE Congress on Evolutionary Computation*, Vancouver, 2006, pp. 1792–1798.

[6] A. Szabo, A. K. F. Prior, and L. N. de Castro, "The proposal of a velocity memoryless clustering swarm," in *Proc. of the 2010 IEEE Congress on Evolutionary Computation*, Barcelona, July 18–23 2010, pp. 1–5.

[7] A. Szabo, L. de Castro, and M. Delgado, "The proposal of a fuzzy clustering algorithm based on particle swarm," in *Proc. of the Third World Congress on Nature and Biologically Inspired Computing (NaBIC)*, oct. 2011, pp. 459–465.

[8] M. Yuwono, S. Su, B. Moulton, and H. Nguyen, "Data clustering using variants of rapid centroid estimation," *IEEE Transactions on Evolutionary Computation*, in press.

[9] T. Wang, "Ca-tree: A hierarchical structure for efficient and scalable coassociation-based cluster ensembles," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 41, no. 3, pp. 686–698, 2011.

[10] T. Wang, "Comparing hard and fuzzy c-means for evidence-accumulation clustering," in *Proceedings of the 18th International Conference on Fuzzy Systems*, ser. FUZZ-IEEE'09. Piscataway, NJ, USA: IEEE Press, 2009, pp. 468–473.

[11] F. Duarte, A. L. N. Fred, A. Lourenco, and M. Rodrigues, "Weighting cluster ensembles in evidence accumulation clustering," in *Artificial intelligence, 2005. epia 2005. portuguese conference on*, 2005, pp. 159–167.

[12] A. Fred and A. Jain, "Combining multiple clusterings using evidence accumulation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 6, pp. 835–850, 2005.

[13] A. Loureno, S. Rota Bul, A. Fred, and M. Pelillo, "Consensus clustering with robust evidence accumulation," in *Energy Minimization Methods in Computer Vision and Pattern Recognition*, ser. Lecture Notes in Computer Science, A. Heyden, F. Kahl, C. Olsson, M. Oskarsson, and X.-C. Tai, Eds. Springer Berlin Heidelberg, 2013, vol. 8081, pp. 307–320.

[14] L. Vendramin, R. J. G. B. Campello, and E. R. Hruschka, "On the comparison of relative clustering validity criteria." in *SIAM International Conference on Data Mining*, 2009, pp. 733–744.

[15] J. Bezdek and N. Pal, "Some new indexes of cluster validity," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 28, no. 3, pp. 301–315, 1998.

[16] C. L. Blake and C. J. Merz. (1998) Uci repository of machine learning databases.

[17] (2007) Image parsing. [Online]. Available: http://www.imageparsing.com/