Data Clustering Using Variants of Rapid Centroid Estimation

Mitchell Yuwono, Steven W. Su, Bruce D. Moulton, and Hung T. Nguyen

Abstract—Prior work suggests that Particle Swarm Clustering (PSC) can be a powerful tool for solving clustering problems. This paper reviews parts of the PSC algorithm, and shows how and why a new class of algorithm is proposed in an attempt to improve on the efficiency and repeatability of PSC. This new implementation is referred to as Rapid Centroid Estimation (RCE). RCE simplifies the update rules of PSC, and greatly reduces computational complexity by enhancing the efficiency of the particle trajectories. On benchmark evaluations with an artificial dataset that has 80 dimensions and a volume of 5000, the RCE variants have iteration times of less than 0.1 seconds, which compares to iteration times of 2 seconds for PSC and modified PSC (mPSC). On UC Irvine (UCI) machine learning benchmark datasets, the RCE variants are much faster than PSC and mPSC, and produce clusters with higher purity and greatly improved optimization speeds. For example, the RCE variants are more than 100 times faster than PSC and mPSC on the UCI breast cancer dataset. It can be concluded that the RCE variants are leaner and faster than PSC and mPSC, and that the new optimization strategies also improve clustering quality and repeatability.

Index Terms—algorithm design and analysis, clustering algorithms, particle swarm optimization, computational complexity.

I. INTRODUCTION

CLUSTERING is used in a broad range of disciplines because it is particularly useful for segmenting large multidimensional data into distinguishable representative clusters [1]–[5]. An objective of clustering is to identify parts of the data that have high degrees of similarity with other parts of the data, and group the similar parts together into clusters. Swarm intelligence has increasingly been used as a tool for solving clustering problems [3]–[8].

Particle Swarm Optimization (PSO) is a stochastic optimization approach originally proposed by Kennedy and Eberhart in 1995 [9]. It uses a model that is inspired by the behavior of flocks of birds [9]. Prior works suggest that PSO algorithms are very capable of solving clustering problems [3]–[7], [9]–[11]. It has been reported that PSO-based solutions are more globally-optimum than conventional methods [5], [7], [9]–[11]. However, in cases where the data has high dimensionality and large volume, PSO based clustering algorithms can become computationally demanding [8], [10]–[14].

Van Der Merwe and Engelbrecht proposed a k-means seeded PSO clustering algorithm in 2003, which was reported with promising results [6]. Van Der Merwe’s approach has been recently adapted to Ever’s regrouping PSO [15] to perform data clustering of head movement and fall signals [3]. A high complexity inherent in the Regrouping PSO clustering algorithm has been observed [3]. These observations were consistent with prior reports that indicate that algorithms that incorporate PSO generally become more computationally expensive when they are applied to problems of higher volume, dimension, and complexity [7], [8], [16]. A recent work by Li and Yao investigated efficient PSO algorithms for large scale optimization [16]. The proposed algorithm uses cooperatively co-evolving multi-swarm PSO, with Cauchy and Gaussian update rules, to effectively solve high dimensional functions with lower computational complexity.

Particle Swarm Clustering (PSC) was proposed by Cohen and de Castro in 2006 [10]. They showed that PSC is superior to K-means for clustering benchmark datasets. In 2010, the Modified PSC (mPSC) was proposed by Szabo in an attempt to reduce the computational complexity of PSC [11]. However, Szabo conceded that the improvement was minimal.

Different swarm algorithms have proposed and experimentally examined in an attempt to develop improved swarm-intelligence clustering methods [3]–[5], [12]–[14]. The methods include an approach termed Rapid Centroid Estimation (RCE) [4], [5], [12]–[14]. One of the distinguishing features of RCE, when compared with its predecessors, is its optimization speed. The RCE update algorithm, when compared with that of PSC, enables RCE to achieve more efficient particle movement and greatly reduced computational complexity [13].

It had previously been observed that standard RCE can stagnate at sub-optimal equilibrium positions depending on the starting positions of its particles, and had previously been proposed two possible strategies for RCE that increase the likelihood of RCE particles exploring different equilibrium positions [14]. These strategies are termed Substitution and Swarm. The superscript “+” is added to RCE (e.g. RCE+) to indicate an RCE algorithm with Substitution strategy, and the prefix Swarm (e.g. Swarm RCE) to indicate an RCE algorithm with Swarm strategy. Swarm RCE+, which uses both strategies, has been recently applied to image clustering problems with promising results, which has been described in previous papers [4], [5]. To discriminate gait cadence and other non-walking activities, Swarm RCE+ was used to cluster...
Discrete Wavelet Transform (DWT) features of spectrogram images from a torso attached inertial measurement unit [4]. For Hartmann-Shack image clustering, it has been reported that Swarm RCE produces clearer and more stable results than conventional thresholding and K-means clustering [5].

In this paper a simplified method for the Substitution strategy is introduced. The new algorithm for Substitution is constructed so that it is more efficient and easier to implement. A new strategy, termed Particle Reset, and a new update scheme, termed White Noise Update are proposed. The authors’ code for RCE has been published on The Mathworks MATLAB website, where it is now available as a free download [17].

This paper is organized as follows. Section I provides an introductory background. Section II presents an overview of PSC and mPSC. Section III presents an overview of RCE and the proposed strategies. Analyses of time complexity, particle behavior and performance on artificial datasets are provided in Section IV. Benchmark test results are given in Section V. Conclusions and future research directions are provided in Section VI.

II. PREDECESSOR ALGORITHM OVERVIEW

A. PSC

Particle Swarm Clustering (PSC) can be viewed as a special modification of PSO, devised specifically for clustering tasks [10]. Conventional PSO clustering assigns each particle to be a representative of a candidate solution [6]. PSC, on the other hand, assigns each particle as a potential centroid candidate [10]. In PSC, the solution is encoded in the whole swarm. In order to partition a dataset with dimension \( n \), in PSC, the solution is encoded in the whole swarm. In order to partition a dataset with dimension \( n \), the number of elements that belong to cluster/particle \( i \), equals 0. In this stagnation scenario particle \( i \) is redirected towards the winning particle \( I_{win} \), a particle representing the cluster with the most members. In this scenario, \( x_i(t) \) is updated immediately in the same iteration using eq. (1) with

\[
X_i(t) = x_{I_{win}}(t) - x_i(t), \quad Y_i(t) = 0, \quad Z_i(t) = 0. \tag{3}
\]

B. mPSC

Szabo proposed a modification to the original PSC called the Modified PSC (mPSC) in 2010 [11]. Szabo proposed that the velocity term can be replaced by \( \Delta x \) which represents a small perturbation in the behavior of the individual. With the definitions of \( X_i(t), Y_i(t), \) and \( Z_i(t) \) unchanged, eq. (1) is modified by defining \( \omega(0) = 0 \) for all \( t > 0 \). The mPSC update method is summarized as follows:

\[
\Delta x_i(t + 1) = \varphi_1 X_i(t) + \varphi_2 Y_i(t) + \varphi_3 Z_i(t), \tag{4}
\]

\[
x_i(t + 1) = x_i(t) + \Delta x_i(t + 1).
\]

At the end of each iteration, there is a case when \( n_i(t) \), the number of elements that belong to cluster/particle \( i \), equals 0. In this stagnation scenario particle \( i \) is redirected towards the winning particle \( I_{win} \), a particle representing the cluster with the most members. In this scenario, \( x_i(t) \) is updated immediately in the same iteration using eq. (1) with

\[
X_i(t) = x_{I_{win}}(t) - x_i(t), \quad Y_i(t) = 0, \quad Z_i(t) = 0. \tag{3}
\]

\[
\text{Algorithm 1} \quad \text{PSC — mPSC}
\]

1. Define the search space \( \Omega = [y^U, y^L] \)
2. \( 1 \leq i \leq n_c \) (number of clusters) : \( x_i = \text{rand}(\Omega), v_i = 0, p_i^f = x_i \)

3. for \( 1 \leq t < t_{max} \) do

4. for \( j = 1 \) to \( n_j \) (data volume) do

5. Calculate distance vector \( d \) between each particle \( x_i \) and data point \( y_j \) using a distance function \( D(\cdot, \cdot) \).

6. \( d_j^i(t) = D(x_i, y_j) : 1 \leq i \leq n_c \)

7. Find particle \( I \), which is closest to the data point \( j \)

8. \( I = \arg \min_{i} d_j^i(t) \)

9. Update \( p_j^i \) and \( g_j \)

10. if \( D(x_i(t), y_j) < D(p_j^i(t), y_j) \) then

11. \( p_j^i(t) = x_i(t) \)

12. end if

13. if \( D(x_i(t), y_j) < D(g_j(t), y_j) \) then

14. \( g_j(t) = x_i(t) \)

15. end if

16. calculate \( X_i(t), Y_i(t), \) and \( Z_i(t) \) using eq. (2)

17. calculate \( x_i(t + 1) \) using eq. (1) for PSC or eq. (4) for mPSC.

18. end for

19. for \( i = 1 \) to \( n_c \) (number of clusters) do

20. if \( n_i(t) == 0 \) then

21. Find winning particle \( I_{win} \)

22. \( X_i(t) = x_{I_{win}}(t) - x_i(t), Y_i(t) = 0, Z_i(t) = 0 \)

23. using eq. (3)

24. calculate \( x_i(t + 1) \) using eq. (1) for PSC or eq. (4) for mPSC.

25. end if

26. end for

27. return \( \{x_i(t_{max}) : 1 \leq i \leq n_c\} \)
II. RAPID CENTROID ESTIMATION

A. Algorithm Overview

RCE is based on the PSC algorithm, but is reconfigured to require less computational complexity, without sacrificing optimization capability [13]. When implementing PSC, computational complexity arises from the frequent calculation of the distance vector $d_i$. A distance matrix computation for a $d$ dimensional problem from $n_c$ particles to $n_j$ input data results in a computational complexity of $O(d \times n_c \times n_j)$ [12].

Based on this finding, a new update rule is designed [12]–[14] such that once. Each position update occurs after all the possible distance terms that are closest to that particle are calculated.

3) A fitness function is used to derive better position combinations. To do this a widely used objective function which minimizes the sum of the intra-cluster distances is implemented.

For each iteration, $x_i(t + 1)$ is updated as follows:

$$\Delta x_i(t + 1) = \frac{\omega(t)\Delta x_i(t)}{K} + \frac{A_i(t) + B_i(t) + C_i(t)}{K},$$

where $A_i(t)$, $B_i(t)$, and $C_i(t)$ indicate collective cognitive, collective social, and collective self-organizing terms, respectively. These terms are calculated as given in eq. (6). $K$ denotes the step resolution constant, $K \geq 1$. $K$ divides the sum of $A_i(t)$, $B_i(t)$, and $C_i(t)$ values into $K$ discrete steps. A larger $K$ value makes RCE particles move in smaller step each position update.

In this paper a new scheme that is different from those described in earlier proposals for RCE is proposed [12]–[14]. This new scheme computes the mean for cognitive, social, and self-organizing terms previously defined in eq. (2) for all input patterns $t$ in cluster $i$ as follows:

$$A_i(t) = \frac{\sum_{j \in C_i} \varphi_{ij} (p_i^j(t) - x_i(t))}{n_i},$$  

$$B_i(t) = \frac{\sum_{j \in C_i} \varphi_{ij} (g_j(t) - x_i(t))}{n_i},$$  

$$C_i(t) = \frac{\sum_{j \in C_i} \varphi_{ij} (y_j - x_i(t))}{n_i},$$

where $\varphi_{ij}$ is a matrix of uniform random numbers $0 \leq \varphi_{ij} \leq 1 \in \mathbb{R}$ that denote an important factor for each term. The $\varphi$ matrix is generated once per iteration.

III. RAPID CENTROID ESTIMATION

The matrix $M(t)$,

$$M(t) = \begin{bmatrix}
    x_1(t) & \cdots & x_{n_c}(t) \\
    (x_{11}(t)) & \cdots & (x_{1n}(t)) \\
    \vdots & \ddots & \vdots \\
    (x_{d1}(t)) & \cdots & (x_{dn}(t))
\end{bmatrix}$$

stores the best position combination which yields the best objective function evaluation.

Different objective functions and distance metrics can be used with RCE; in this paper an objective function is defined as to minimize the sum of intra-cluster distance or more widely known as sum of squared distance (SSD) [18] as follows:

$$f(x, y) = \sum_{i=1}^{n_c} \sum_{y_j \in C_i} D(x_i, y_j),$$

where $C_i$ denotes the cluster set $i$ with the centroid $x_i$, $D(\cdot, \cdot)$ denotes the specified distance function.

B. Optimization Strategies

PSC-based algorithms can stagnate on a specific equilibrium position depending on the starting coordinates of the particles [14]. In RCE, it is proposed that it is not ideal for a search to stagnate in a specific minimum because this position might be a local minimum. One of the challenges was to implement a strategy that would gracefully break an equilibrium state.

1) New Substitution Strategy: A new Substitution strategy is proposed in order to simplify prior implementation of the strategy [14]. The purpose of the Substitution strategy is to force particles in a search space to reach alternate equilibrium positions by introducing position instability. After each position update episode for a particle $i$, apply

$$x_i(t + 1) = \begin{cases}
    x_i(t + 1) + N(0, \sigma) & \text{if } \varphi \leq \frac{\varepsilon}{K}, \\
    x_i(t + 1) & \text{otherwise}
\end{cases}$$

$$\Delta x_i(t + 1) = \begin{cases}
    0 & \text{if } \varphi \leq \frac{\varepsilon}{K}, \\
    \Delta x_i(t + 1) & \text{otherwise}
\end{cases}$$

where $\varphi$ is a uniform random number $0 \leq \varphi \leq 1 \in \mathbb{R}$, $x_i$ is the position of the winning particle, and $N(0, \sigma)$ is a Gaussian random vector with mean $\mu = 0$ and standard deviation $\sigma$ of each dimension of the data being clustered. $\varepsilon$ denotes the Substitution probability parameter. Larger $\varepsilon/K$ increases the substitution frequency. Optimal $\varepsilon/K$ values lie between $0.01 \leq \varepsilon/K \leq 0.05$. Performance degradation is apparent with $\varepsilon/K > 0.05$. A graph showing the final objective function evaluation with varying $\varepsilon/K$ values on different datasets averaged over 80 trials is shown in Figure 1.

2) Particle Reset: The Particle Reset strategy is triggered when group fitness $f(M(t), y)$ does not improve after a number of iterations. Stagnation can be detected using a stagnation counter $\delta$ which is updated as follows:

$$\delta(t + 1) = \begin{cases}
    \delta(t) + 1 & \text{if } f(x(t), y) \geq f(M(t), y), \\
    0 & \text{if } f(x(t), y) < f(M(t), y)
\end{cases}$$

Copyright (c) 2013 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.
When \( \delta(t+1) > \delta_{\text{max}} \) this strategy reinitializes all particles in the group without resetting the best position matrix \( M(t) \). Values being reinitialized are only \( x_i(t) \) and \( \Delta x_i(t) \). Swarm convergence is detected when \( f(M(t), y) \) does not improve even after many resets. The superscript \( r \) (e.g. \( \text{RCE}^r \)) denotes an RCE with Particle Reset strategy.

3) Swarm Strategy: A limitation inherited from PSC is that the number of particles in a group is fixed according to the desired number of clusters. To overcome this limitation, a strategy is proposed that is intended to handle increases in swarm size, without increasing the number of clusters \([14]\). An RCE group consists of \( n_c \) particles, each corresponding to a centroid prototype.

A Swarm\(\{n_m\}\) RCE consists of \( n_m \) RCE groups working in parallel. For example, Swarm\(\{3\}\) RCE indicates a centroid optimization using 3 RCE groups, while Swarm\(\{5\}\) RCE indicates a centroid optimization using 5 RCE groups.

Each RCE group \( \text{RCE}\{n\} \) stores a best position matrix \( M^n(t) \) such as defined in eq. \((7)\). The swarm strategy communicates each \( M^n(t) \) such that the potentially optimal positions are informed to all RCE groups.

On the start of every iteration, each group in the swarm contributes by sharing its minimum matrix \( M_n(t) \) such that

\[
M^D(t) = \left( \left[ M_{d \times n_c}^1(t) \right] \left[ M_{d \times n_c}^2(t) \right] \cdots \left[ M_{d \times n_c}^{n_m}(t) \right] \right).
\]

The matrix \( M^D \) has \( n_c \times n_m \) columns denoting the number of centroid vectors stored in \( M^D \). When using the Swarm strategy, the term collective minimum,

\[
D_i(t) = \frac{\sum_{\forall M^n(t) \in x_i(t)} \varphi_{i,n} \left( M^n(t) - x_i(t) \right)}{n_i}.
\]

is added to eq. \((6)\).

The swarm update rule in eq. \((12)\) includes \( D_i(t) \) such that

\[
\Delta x_i(t+1) = \omega(t) \Delta x_i(t) + A_i(t) + B_i(t) + C_i(t) + D_i(t),
\]

\[
\Delta x_i(t+1) \in [-\Delta x_{\text{max}}, +\Delta x_{\text{max}}],
\]

\[
x_i(t+1) = x_i(t) + \Delta x_i(t+1).
\]

4) White Noise Update Scheme: Random number generation also contributes to the increase in time complexity. The number of random matrices generated in the standard RCE is dependent on the number of particles.

The employment of noise in optimization algorithms is not a new concept. It has been used in algorithms such as Genetic Algorithm \([20]\) and Differential Evolution \([21]\). These methods have been shown to be an effective method for nonlinear stochastic search. Inspired by these works a new method is proposed. This method allows RCE to generate a matrix of random number with fixed size, independent of the number of particles. This alteration to the update scheme further reduces the complexity of the RCE update rule. In this scheme uncorrelated white noise is injected into each datum \( y_j \),

\[
\tilde{y}_j(t) = y_j + w_j(t) : \forall j,
\]

where \( \tilde{y}_j(t) \) denotes datum coordinate \( y_j \) injected with white noise \( w_j(t) \). \( w_j(t) \) is a uniform random matrix with size of \( d \times n_j \), and for each dimension \( w_{d,j}(t) \in [-\sigma_d, +\sigma_d] \). Due to the constant coordinate change in the information domain, the complexity of the RCE update function can be further reduced so that for each iteration, only one matrix of random numbers is generated regardless of the number of particles. Consequently, \( A_i(t), B_i(t), C_i(t), \) and \( D_i(t) \) can

---

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.
be calculated as follows:

\[
A_i(t) = \sum_{\forall y_j \in \Omega_i(t)} \frac{n_i}{\sum_{\forall y_j \in \Omega_i(t)} n_i} p_i^j(t) - x_i(t),
\]

\[
B_i(t) = \sum_{\forall y_j \in \Omega_i(t)} \frac{n_i}{\sum_{\forall y_j \in \Omega_i(t)} n_i} g_j(t) - x_i(t),
\]

\[
C_i(t) = \sum_{\forall y_j \in \Omega_i(t)} \frac{n_i}{\sum_{\forall y_j \in \Omega_i(t)} n_i} \tilde{\gamma}_j(t) - x_i(t),
\]

\[
D_i(t) = \sum_{\forall M^D(1) \subset \Omega_i(t)} \frac{n_i}{\sum_{\forall M^D(1) \subset \Omega_i(t)} n_i} M^D(t) - x_i(t),
\]

without the need for a uniform random matrix multiplicand \(\varphi_{i,j}\).

The pseudocode for the RCE algorithm with the Particle Reset, Substitution, Swarm, and White Noise Update strategies enabled is presented in Algorithm 2.

IV. Performance Analysis

A. Particle Behavior

The performance of each algorithm was evaluated using a two class well separated Gaussian dataset in two dimensional space [13]. The dataset consisted of 250 normally distributed data points with mean of \([0.5 0.5]\) and standard deviation of \([0.15 0.15]\), and 250 normally distributed data points with mean of \([-0.5 -0.5]\) and standard deviation of \([0.15 0.15]\). The result of this experiment is presented in Figure 2. Comparing the particle behaviors in Figure 2a and 2b, PSC and mPSC share similar trajectory characteristics. RCE shows particle movements that are more effective than the movements of PSC and mPSC [13]. RCE with both the Standard Update rule (Figure 2c) and the White Noise Update rule (Figure 2d) resulted in smoother trajectories and significantly lower numbers of movements towards particle stagnation than the predecessors.

B. Centroid Optimization Performance

Substitution, Particle Reset and Swarm strategies force the particles to avoid stagnation and more actively explore the search space as seen in Figure 2e and 2f. These strategies increase the chance of finding better centroid locations due to a greater number of attempts to escape local minima. A visual comparison of the sum of intra-cluster distance evolution for K-means, PSC, mPSC, RCE, RCE\(^\ast\), RCE\(^\ast\ast\), and Swarm [5] RCE\(^\ast\ast\) on a clustering process of the UCI datasets [19] is shown in Figure 3. For the iris, glass, wine, cancer, and diabetes datasets, the RCE\(^\ast\ast\) variants achieve the lowest SSD compared to the other algorithms. For both the optical digits and musk datasets, mPSC achieve the lowest SSD.

C. Iteration Speed

The iteration speed of each algorithm was evaluated using Gaussian synthetic datasets that each have three classes. For each dataset, the dimensionality was gradually increased from 1 to 80 and the volume was gradually increased from 100 to 5000.

The experimental results in Figure 4 show that PSC and mPSC share similar trends in iteration time, while standard RCE and White Noise RCE are capable of achieving much smaller iteration times. This is attributed to the reduced per-iteration computational complexity and stabilized particle movement [12]. Comparisons of the iteration times for PSC, mPSC, RCE (Standard Update), and RCE (White Noise Update) are given in Figure 4a, 4b, 4c, and 4d, respectively.

In this experiment an \(\alpha\) value is defined which indicates the gradient of iteration time increase to volume increase \(\partial t / \partial (n_j)\). For example, an \(\alpha\) value of 4e-3 indicates that for each unit increase of volume, the iteration time of the particular algorithm will increase by 4 milliseconds.

The \(\alpha\) values for PSC and mPSC with respect to \(n_c\) are given in Figure 5a. The \(\alpha\) values for standard and White Noise RCE with respect to \(n_c\) are given in Figure 5b. The
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

Figure 2: Comparison of particle trajectories associated with the different algorithms. PSC particle trajectories with $v_{\text{max}} = 0.01 \Omega$ are shown in Figure 2a. mPSC particle trajectories with $0 \leq \varphi_{1,2,3} \leq 0.1$ are shown in Figure 2b. Standard RCE and White Noise RCE (Figure 2c and 2d) behave similarly to one another. Both algorithms result in trajectories that are more efficient than those of the predecessors. The Substitution, Particle Reset, and Swarm Strategies cause RCE particles to explore different position combinations (Figure 2e and 2f).

Figure 3: $f(x, y)$ evolution of K-means, PSC, mPSC, RCE, RCE$^+$, and Swarm{5} RCE$^+$ on various UCI benchmark datasets [19].

relationships between $\alpha$ value, swarm size, and number of clusters are given in Figure 5c. The data that is given in Table I are based on the plots that are depicted in Figures 4 and 5.

**TABLE I: Fitted polynomial model for iteration times of various algorithms on generated dataset**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Fitted Polynomial Model</th>
<th>Coefficients$^*$</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSC</td>
<td>$t(n_j, d, n_c) = \alpha d n_j$</td>
<td>$\alpha = k_1 + k_2 n_c$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$k_1 = 1.88e-4$, $k_2 = 1.85e-5$</td>
</tr>
<tr>
<td>mPSC</td>
<td>$t(n_j, d, n_c) = \alpha d n_j$</td>
<td>$\alpha = k_1 + k_2 n_c$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$k_1 = 1.75e-4$, $k_2 = 1.84e-5$</td>
</tr>
<tr>
<td>RCE, RCE$^+$, RCE$^{+\oplus}$</td>
<td>$t(n_j, d, n_c) = \alpha d n_j$</td>
<td>$\alpha = k_1 + k_2 n_c$</td>
</tr>
<tr>
<td>(Standard Update)</td>
<td></td>
<td>$k_1 = 9.3e-8$, $k_2 = 1.44e-8$</td>
</tr>
<tr>
<td>Swarm${n_m}$ RCE$^{+\oplus}$</td>
<td>$t(n_j, d, n_c, n_m) = \alpha d n_j$</td>
<td>$\alpha = (k_1 + k_2 n_c) n_m$</td>
</tr>
<tr>
<td>(Standard Update)</td>
<td></td>
<td>$k_1 = 9.8e-8$, $k_2 = 1.6e-8$</td>
</tr>
<tr>
<td>RCE, RCE$^+$, RCE$^{+\oplus}$</td>
<td>$t(n_j, d, n_c) = \alpha d n_j$</td>
<td>$\alpha = k_1 + k_2 n_c$</td>
</tr>
<tr>
<td>(White Noise Update)</td>
<td></td>
<td>$k_1 = 6.8e-8$, $k_2 = 1.43e-8$</td>
</tr>
<tr>
<td>Swarm${n_m}$ RCE$^{+\oplus}$</td>
<td>$t(n_j, d, n_c, n_m) = \alpha d n_j$</td>
<td>$\alpha = (k_1 + k_2 n_c) n_m$</td>
</tr>
<tr>
<td>(White Noise Update)</td>
<td></td>
<td>$k_1 = 6.1e-8$, $k_2 = 1.6e-8$</td>
</tr>
</tbody>
</table>

$^*$Constant values vary depending on the time complexity to calculate the distance matrix used in the optimization. In this table these coefficient values are measured from optimizations using Squared Euclidean Distance metric.

The curve fitting results in Table I show the effects of the different algorithms. The coefficients are obtained from least squares approximations of the experimental results in Figures 4 and 5. The distance metric used is *Squared Euclidean*.

The PSC and mPSC algorithms share a similar update rule, which operates element by element. According to the proposed algorithms, for each element, the distance between the particular datum to each particle is calculated, producing a vector with length of $n_c$ denoting its distance to each particle. Consequently, as the size of the distance matrix grows, an increase in $\alpha$ value up to 188 microseconds is observed as each additional element is calculated. According to the $\alpha$ trend shown in Figure 5a, mPSC has slightly lower $\alpha$ value (1.75e-4) as compared to PSC (1.88e-4).

The RCE update scheme reduces the computational complexity by performing a batch distance calculation at the start of each iteration. On each iteration, each RCE particle makes a decisive movement to the centre of mass of each cluster based on the update rule described in eq. (5). When the $\alpha$ values for the different algorithms are compared, RCE algorithms have $\alpha$ values at least 2000/d times smaller than those of the PSC and mPSC algorithms as shown in Table I, Figure 5a and 5b. Figure 5b shows that the new *White Noise Update* scheme further simplifies the standard RCE update scheme by reducing the frequency of random matrix generation, allowing an $\alpha$ value improvement of 61 nanoseconds for each unit increase.

*Copyright (c) 2013 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.*
of $d \times n_j$.

Table I also shows the relationship between RCE swarm size and iteration time. In Table I, the $\alpha$ gradient denotes the gradient of iteration time increase per unit increase of $d \times n_j$. It is shown that $\alpha$ is directly proportional to $n_c$ and $n_m$. White noise RCE has an $\alpha$ gradient that is smaller than that of standard RCE. This means that the new White Noise Update scheme is preferable when the Swarm strategy is used for larger data.

V. PERFORMANCE ON BENCHMARK DATASETS

The performance of the algorithms was benchmarked using five openly available datasets from the UCI machine learning repository [19]. The datasets used are the Iris, Glass, Wine, Breast Cancer, Diabetes Pima Indian, Optical Digits segmentation, and Musk Molecules datasets. Characteristics of these datasets are shown in Table II. The clustering process was repeated 80 times. The experimental results are presented in Table IV. A comparison of the time taken by each algorithm is given in Table III. Box-plots comparisons of the resulting centroid purities, CH indices, and optimization times of the different algorithms on benchmark datasets are provided in Figures 8 to 10.

Table II: The datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Objects</th>
<th>Attributes</th>
<th>Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iris</td>
<td>150</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Glass</td>
<td>215</td>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>Wine</td>
<td>178</td>
<td>13</td>
<td>3</td>
</tr>
<tr>
<td>Breast Cancer</td>
<td>699</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>Diabetes</td>
<td>768</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>Optical Digits</td>
<td>5620</td>
<td>64</td>
<td>10</td>
</tr>
<tr>
<td>Musk (version 1 &amp; 2)</td>
<td>7074</td>
<td>167</td>
<td>2</td>
</tr>
</tbody>
</table>

PSC $v_{max}$ is set to 0.001, and the mPSC random range is set to $0 \leq \varphi_{1,2,3} \leq 0.1$. In this experiment, RCE uses the White Noise Update scheme. The parameters for RCE were set as follows. Swarm size was set to 5 ($n_m = 5$), $K = 12$, $\epsilon/K = 0.03$, $\delta_{max} = 200$, $\Delta x_{max} = 0.01 \Omega$, and the maximum iteration was set to 500. The speed is measured using iteration time ($T_i$) and stopping time ($\tau$), where $\tau$ is the total optimization time in seconds to reach a local minimum solution. $\tau$ is calculated as follows:

$$\tau = t_{stop} \times T_i,$$  \hspace{1cm} (15)

where $t_{stop}$ is the number of iterations needed to reach a local optimum solution.

The performance of the clustering algorithms is measured using Entropy (E), Purity (P), and Percentage Misclassification (Pm) as follows:

$$E = -\frac{1}{n_j} \sum_{r=1}^{n_c} \sum_{i=1}^{k} n_r^i \ln \frac{n_r^i}{n_r},$$  \hspace{1cm} (16)

$$P = \frac{1}{n_j} \sum_{r=1}^{n_c} \max(n_r^i),$$  \hspace{1cm} (17)

$$Pm = \frac{n_j^{fp}}{n_j},$$  \hspace{1cm} (18)

where $r$ indicates the cluster index, $k$ indicates the total number of classes in cluster $r$, $n_r$ indicates the number of elements in the cluster $r$, $n_r^i$ indicates the number of elements of class $i$ inside the cluster $r$, $n_j$ indicates volume of the dataset, and $n_j^{fp}$ indicates the number of false positive detections.

Entropy (E) gives an indication of cluster homogeneity. Lower entropy indicates that objects in the database are homogenous. Purity (P) is used to give an indication of the purity of the cluster by taking the ratio of the dominant class of the group in relation to the total number of objects inside the group. In this experiment, it is the weighted sum of correct classifications. High purity is desirable for a good cluster. The Percent Misclassified (Pm) is the ratio of false positive classifications to the number of objects. Low Pm suggests good clusters.

In addition to E, P, and Pm, validity indices of the clustering results are important measures for comparing the clustering performances of different clustering algorithms [18], [22]. The cluster validity indices used in this experiment are the Dunn index [23], adjusted rand index [24], and Calinski-Harabasz (CH) index [25].

The Dunn index [23] is used to identify clusters that are compact and well separated [22]. The Dunn index is formulated as follows:

$$Dunn = \min_{i=1,...,n_c} \left( \min_{j=1,...,n_c,j\neq i} \left( \frac{\min D(y_x \in x_r, y_y \in x_r)}{\max D(y_x \in x_r, y_y \in x_r)} \right) \right).$$  \hspace{1cm} (19)

A high Dunn index suggest that the clustering result produces clusters that are compact and well separated.

The adjusted rand index [24] measures the agreement between elements in clustered set $C$ and elements in target set $T$ [22]. The Adjusted Rand Index is calculated as follows:

$$Adj. R = \frac{\binom{n_j}{2}(a + d) - ((a + b)(a + c) + (c + d)(b + d))}{\binom{n_j}{2} - ((a + b)(a + c) + (c + d)(b + d))},$$  \hspace{1cm} (20)

where:

- $a$: number of element pairs belong to the same set in both $C$ and $T$,
- $b$: number of element pairs that belong to the same set in $C$ but different sets in $T$,
- $c$: number of element pairs that belong to different sets in $C$ but the same set in $T$, and
- $d$: number of element pairs that belong to different sets in both $C$ and $T$.

An adjusted rand index of one indicates a perfect agreement, zero suggests agreement due to chance, while negative indicates agreement less than chance [22].

The CH index [25] measures clustering quality based on the traces of between-cluster and within-cluster distance scatter.
matrices. CH index is formulated as follows:

$$
m = \frac{\sum_{j=1}^{n_j} y_j}{n_j},$$

$$Tr(S_B) = \sum_{i=1}^{n} \sum_{j=1}^{n_i} n_i D(x_i, m),$$

$$Tr(S_W) = \sum_{i=1}^{n} \sum_{j=1}^{n} D(y_j, x_i),$$

$$CH = \frac{Tr(S_B)/(n_c - 1)}{Tr(S_W)/(n_j - n_c)},$$

where \(D(\cdot, \cdot)\) denotes the specified distance function. A large value of CH suggests a clustering result with good quality [22].

The results given in Table IV indicate that the performance of the RCE algorithms was better than that of the predecessors. It can be seen that when using the White Noise Update rule with both Substitution and Particle Reset strategies, RCE\(^+\) produces solutions with levels of purity and repeatability that are, in most cases, substantially greater than those of the other algorithms. It is also seen that performance is further improved with the Swarm strategy. In all datasets, Swarm\([5]\) RCE\(^+\) achieves the highest purity centroid locations. PSC and mPSC also show relatively good performance for all benchmark datasets, however the time taken by PSC and mPSC to achieve those results was in most cases several orders of magnitude larger. Table III shows that the new algorithms are significantly faster than the predecessors. For the breast cancer dataset, the iteration times of RCE\(^+\), and Swarm\([5]\) RCE\(^+\) compared to PSC are 139.67 and 27.5 times faster, respectively.

**TABLE III: Percentage improvement in iteration time (Ti) relative to PSC**

<table>
<thead>
<tr>
<th>Dataset</th>
<th>mPSC</th>
<th>RCE</th>
<th>RCE(^+)</th>
<th>Swarm([5])</th>
<th>RCE(^+)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iris</td>
<td>2.6%</td>
<td>5978.3%</td>
<td>5909.8%</td>
<td>813.5%</td>
<td></td>
</tr>
<tr>
<td>Glass</td>
<td>3.7%</td>
<td>2779.4%</td>
<td>2777.4%</td>
<td>383%</td>
<td></td>
</tr>
<tr>
<td>Wine</td>
<td>4.3%</td>
<td>4963.5%</td>
<td>4857.2%</td>
<td>752.9%</td>
<td></td>
</tr>
<tr>
<td>Breast Cancer</td>
<td>4.78%</td>
<td>14311.7%</td>
<td>13967.2%</td>
<td>2750.7%</td>
<td></td>
</tr>
<tr>
<td>Diabetes</td>
<td>5.1%</td>
<td>9915.5%</td>
<td>10012.9%</td>
<td>1828.3%</td>
<td></td>
</tr>
<tr>
<td>Optical Digits</td>
<td>3.5%</td>
<td>2734.3%</td>
<td>2723.5%</td>
<td>897.9%</td>
<td></td>
</tr>
<tr>
<td>Musk</td>
<td>7.5%</td>
<td>1332.2%</td>
<td>1302.9%</td>
<td>380.1%</td>
<td></td>
</tr>
</tbody>
</table>

*improvement \% = ((t\(_{PSC}^{A}/t\(_{algorithm}^{B}) - 1) \times 100%\)

In order to analyze the experimental results regarding the time complexity of each algorithm, a statistical significance test using one-way ANOVA is performed based on the experimental result collected from centroid optimization of the optical digits and musk molecules datasets using the null hypothesis, \(H_0\): there is no difference between iteration time of algorithm A and algorithm B. The tests were done using the MATLAB statistics toolbox.

Similarly, the resulting CH cluster validity indices for each algorithm on each dataset were analyzed using the null hypothesis, \(H_0\): there is no difference between the resulting CH indices of algorithm A and algorithm B.

In all cases, \(H_0\) is rejected when the p-value is \(\leq 0.05\). The p-values from the ANOVA tests on the iteration time of optical digits and musk molecules datasets are shown in Figure 6. The p-values from the ANOVA tests on the resulting CH cluster validity indices on all benchmark datasets are shown in Figure 7. The box-plots are shown in Figure 9.

**Figure 6:** P-values to three decimal places based on the iteration times given in Table IV. Grey boxes indicate \(p \leq 0.05\). I. K-means; II. PSC; III. mPSC; IV. RCE; V. RCE\(^+\); VI. Swarm\([5]\) RCE\(^+\).

The p-values in Figure 6 indicate that the differences in iteration times between all of the pairs of algorithms, except between RCE and RCE\(^+\), are statistically significant.

Figure 8 shows box-plots comparisons for purity (P) from Table IV. Figure 8 shows that the RCE\(^+\) variants are able to achieve results that are more consistent than the results of K-means, PSC, mPSC, and RCE, with Swarm\([5]\) RCE\(^+\) being most consistent.

Figure 9 shows box-plot comparisons for CH indices that are given in Table IV. Both Table IV and Figure 9 show that the RCE\(^+\) variants produce clusters that have CH indices that are, on average, higher than those of K-means, PSC, mPSC, and RCE on the iris, wine, diabetes, optical digits, and musk molecules datasets. Observing the ANOVA results of the CH indices on the iris, wine, diabetes, and optical digits datasets (Figures 7a, 7c, 7e, and 7f), the clusters produced by PSC, mPSC and RCE are not significantly different from one another. In addition, RCE\(^+\) and Swarm\([5]\) RCE\(^+\) produce clusters that are not significantly different from each other for the glass, wine, diabetes and musk molecules datasets (Figures 7b, 7c, 7e, and 7g).

Figure 10 shows box-plot comparisons for optimization/stopping times (\(\tau\)) that are given in Table IV. In terms of optimization time, Table IV and Figure 10 show that k-means is still the best algorithm, followed by RCE and RCE\(^+\). Swarm\([5]\) RCE\(^+\) has a longer optimization time than the other RCE variants. PSC and mPSC exhibit the longest optimization times.

**VI. CONCLUSIONS AND FUTURE DIRECTIONS**

This paper describes and tests variations of an algorithm that is referred to as the Rapid Centroid Estimation (RCE). The paper presents a comparative study of the results of the Particle Swarm Clustering (PSC), modified Particle Swarm Clustering (mPSC) and RCE algorithms on different datasets, and using different strategies. The results suggest that the RCE variants are faster, produce better clusters, and are potentially useful for...
<table>
<thead>
<tr>
<th>Dataset</th>
<th>Metric ‡</th>
<th>K-means</th>
<th>PSC</th>
<th>mPSC</th>
<th>RCE</th>
<th>RCE⁺⁺</th>
<th>Swarm {5} RCE⁺⁺</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iris (Correlation)</td>
<td>Adj.</td>
<td>0.16 ± 0.03</td>
<td>0.31 ± 0.16</td>
<td>0.32 ± 0.09</td>
<td>0.16 ± 0.17</td>
<td>0.16 ± 0.17</td>
<td>0.15 ± 0.02</td>
</tr>
<tr>
<td></td>
<td>0.5%</td>
<td>4.66%</td>
<td>6.26%</td>
<td>8.67%</td>
<td>8.1%</td>
<td>8.1%</td>
<td>4.9%</td>
</tr>
<tr>
<td></td>
<td>1.2e-3</td>
<td>3.3e-3</td>
<td>3.8e-3</td>
<td>4.2e-3</td>
<td>0.15</td>
<td>0.15</td>
<td>0.15</td>
</tr>
<tr>
<td></td>
<td>±0.012</td>
<td>0.626 ± 0.184</td>
<td>0.584 ± 0.154</td>
<td>0.579 ± 0.164</td>
<td>0.75 ± 0.15</td>
<td>0.79 ± 0.11</td>
<td>0.18 ± 0.18</td>
</tr>
<tr>
<td></td>
<td>±0.013</td>
<td>0.31 ± 0.017</td>
<td>0.5 ± 0.015</td>
<td>0.58 ± 0.009</td>
<td>0.50 ± 0.019</td>
<td>0.50 ± 0.019</td>
<td>0.18 ± 0.015</td>
</tr>
<tr>
<td></td>
<td>0.3%</td>
<td>6.1%</td>
<td>8.2%</td>
<td>8.8%</td>
<td>8.1%</td>
<td>8.1%</td>
<td>4.9%</td>
</tr>
<tr>
<td></td>
<td>±0.16</td>
<td>0.84 ± 0.013</td>
<td>1.2 ± 0.013</td>
<td>1.6 ± 0.016</td>
<td>2.3 ± 0.016</td>
<td>2.3 ± 0.016</td>
<td>2.3 ± 0.016</td>
</tr>
<tr>
<td></td>
<td>±0.017</td>
<td>2.34 ± 0.012</td>
<td>3.3e ± 0.013</td>
<td>4.3e ± 0.013</td>
<td>4.3e ± 0.013</td>
<td>4.3e ± 0.013</td>
<td>4.3e ± 0.013</td>
</tr>
<tr>
<td></td>
<td>0.5%</td>
<td>9.6%</td>
<td>12%</td>
<td>15%</td>
<td>15%</td>
<td>15%</td>
<td>4.9%</td>
</tr>
<tr>
<td></td>
<td>±0.018</td>
<td>0.62 ± 0.013</td>
<td>0.84 ± 0.013</td>
<td>1.2e ± 0.013</td>
<td>1.6e ± 0.016</td>
<td>1.6e ± 0.016</td>
<td>1.6e ± 0.016</td>
</tr>
<tr>
<td></td>
<td>±0.019</td>
<td>1.2 ± 0.013</td>
<td>1.6 ± 0.016</td>
<td>2.0 ± 0.016</td>
<td>2.3 ± 0.016</td>
<td>2.3 ± 0.016</td>
<td>2.3 ± 0.016</td>
</tr>
<tr>
<td></td>
<td>0.5%</td>
<td>9.6%</td>
<td>12%</td>
<td>15%</td>
<td>15%</td>
<td>15%</td>
<td>4.9%</td>
</tr>
<tr>
<td></td>
<td>±0.020</td>
<td>0.62 ± 0.013</td>
<td>0.84 ± 0.013</td>
<td>1.2e ± 0.013</td>
<td>1.6e ± 0.016</td>
<td>1.6e ± 0.016</td>
<td>1.6e ± 0.016</td>
</tr>
<tr>
<td></td>
<td>±0.021</td>
<td>1.2 ± 0.013</td>
<td>1.6 ± 0.016</td>
<td>2.0 ± 0.016</td>
<td>2.3 ± 0.016</td>
<td>2.3 ± 0.016</td>
<td>2.3 ± 0.016</td>
</tr>
<tr>
<td></td>
<td>0.5%</td>
<td>9.6%</td>
<td>12%</td>
<td>15%</td>
<td>15%</td>
<td>15%</td>
<td>4.9%</td>
</tr>
<tr>
<td></td>
<td>±0.022</td>
<td>0.62 ± 0.013</td>
<td>0.84 ± 0.013</td>
<td>1.2e ± 0.013</td>
<td>1.6e ± 0.016</td>
<td>1.6e ± 0.016</td>
<td>1.6e ± 0.016</td>
</tr>
<tr>
<td></td>
<td>±0.023</td>
<td>1.2 ± 0.013</td>
<td>1.6 ± 0.016</td>
<td>2.0 ± 0.016</td>
<td>2.3 ± 0.016</td>
<td>2.3 ± 0.016</td>
<td>2.3 ± 0.016</td>
</tr>
</tbody>
</table>

*Bold fonts indicates algorithm with the best performance metric.

1Bold fonts indicates algorithm with the best performance metric.

2Performance metrics are calculated according to eqs. (15) to (21).

3Measured values correspond to mean (μ) ± standard deviation (σ) of the experimental results over 80 trials.
In addition, further work is needed to test RCE using a wider range of clustering problems. The current RCE algorithm is likely that further work would improve the efficiency of RCE.

A limitation of the current algorithm is that the number of clusters is assumed to be known beforehand. Further work is needed to develop and improve this algorithm to include an entropy minimization objective, which would allow the algorithm to find the optimum number of clusters itself [29]. The feasibility of the RCE algorithm for performing clustering using combinations of objective functions [30], cluster validity indices [22], and information theoretic measures [31] will be investigated.

finding solutions in large datasets of high dimensionality. The Swarm, Particle Reset and Substitution strategies are shown to increase the repeatability of the algorithm.

Notwithstanding the encouraging results to date, it seems likely that further work would improve the efficiency of RCE.
Figure 9: Box-plots [26], [27] of CH indices given in Table IV. x-axis indicates algorithms: I. K-means; II. PSC; III. mPSC; IV. RCE; V. RCE$^+$; VI. Swarm$^+$.

Figure 10: Box-plots [26], [27] of optimization times given in Table IV. x-axis indicates algorithms: I. K-means; II. PSC; III. mPSC; IV. RCE; V. RCE$^+$; VI. Swarm$^+$.

REFERENCES


