

RESEARCH ARTICLE

# Accelerated Simplified Swarm Optimization with Exploitation Search Scheme for Data Clustering

Wei-Chang Yeh<sup>‡</sup>, Chyh-Ming Lai<sup>‡\*</sup>

Department of Industrial Engineering and Engineering Management, National Tsing Hua University, Hsinchu City, Taiwan

<sup>‡</sup> Current address: No. 101, Section 2, Kuang-Fu Road, Hsinchu, Taiwan 30013, R.O.C.

\* [chyh.ming.lai@gmail.com](mailto:chyh.ming.lai@gmail.com)

## Abstract

Data clustering is commonly employed in many disciplines. The aim of clustering is to partition a set of data into clusters, in which objects within the same cluster are similar and dissimilar to other objects that belong to different clusters. Over the past decade, the evolutionary algorithm has been commonly used to solve clustering problems. This study presents a novel algorithm based on simplified swarm optimization, an emerging population-based stochastic optimization approach with the advantages of simplicity, efficiency, and flexibility. This approach combines variable vibrating search (VVS) and rapid centralized strategy (RCS) in dealing with clustering problem. VVS is an exploitation search scheme that can refine the quality of solutions by searching the extreme points nearby the global best position. RCS is developed to accelerate the convergence rate of the algorithm by using the arithmetic average. To empirically evaluate the performance of the proposed algorithm, experiments are examined using 12 benchmark datasets, and corresponding results are compared with recent works. Results of statistical analysis indicate that the proposed algorithm is competitive in terms of the quality of solutions.



## OPEN ACCESS

**Citation:** Yeh W-C, Lai C-M (2015) Accelerated Simplified Swarm Optimization with Exploitation Search Scheme for Data Clustering. PLoS ONE 10 (9): e0137246. doi:10.1371/journal.pone.0137246

**Editor:** Zhaohong Deng, Jiangnan University, CHINA

**Received:** April 21, 2015

**Accepted:** August 14, 2015

**Published:** September 8, 2015

**Copyright:** © 2015 Yeh, Lai. This is an open access article distributed under the terms of the [Creative Commons Attribution License](https://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

**Data Availability Statement:** All data files are available from the UCI Machine Learning Repository (<http://archive.ics.uci.edu/ml>).

**Funding:** The authors have no support or funding to report.

**Competing Interests:** The authors have declared that no competing interests exist.

## Introduction

Cluster analysis is principally used to explore useful knowledge, particularly inherent structure, hidden in a dataset with scarce pre-existent information. Such technique has been frequently applied in many complicated tasks, including pattern recognition, image analysis, facility location, and other fields of engineering [1–2]. Cluster analysis aims to categorize unlabeled data into different clusters on the basis of the similarity between data instances. Similarity is generally measured by the distance metric and can also be treated as an optimization problem that requires an optimal assignment of objects to clusters by minimizing the sum of distance metric between each object and its cluster centroid [3].

K-means (KM) clustering, which uses distance metric to partition data into  $K$  clusters, is common and fundamental because of its simplicity and efficiency. However, the initial state

may cause the algorithm to be trapped in local optima, thereby affecting the quality of the solution [4]. Recent studies have made significant progress in overcoming the drawback of KM clustering, particularly by using evolutionary algorithms, including genetic algorithm [5–6], tabu search approach [7], ant colony optimization [8–9], artificial bee colony algorithm [10], and particle swarm optimization [11–13]. Several algorithms inspired by physical phenomenon have also been employed. These algorithms include simulated annealing algorithm [14], big bang-big crunch optimization [15], gravitational search algorithm combined with KM (GSA-KM) [16], and black hole algorithm (BH) [3].

Those approaches may take time to converge in dealing with clustering problems, particularly large problems, because the initial particles are randomly generated and the subsequent updates are probabilistic. To speed up the convergence rate, Krishna and Murty [5] have employed an accelerated strategy called KM operator (KMO), which merges the principles of KM into the clustering algorithm to efficiently find an effective solution. KMO uses arithmetic average to determine new cluster centers in each generation. However, KMO may be disabled after a number of generations. In this case, the computation cost is wasted, and the diversity of population is restricted by the arithmetic average. Some clustering algorithms have combined with the KM by using the result of KM as one of the initial solutions to speed up the convergence rate [9, 13, 16]. However, the outcome of KM heavily depends on the initial choice of the cluster centers and may converge to the local optima rather than global optima. As a result, these hybrid algorithms may start searching from a local optimum and obtain poor-quality solutions.

To overcome such problem, this study proposes an improved simplified swarm optimization (SSO) that combines variable vibrating search (VVS) and rapid centralized strategy (RCS) to solve clustering problems. VVS is an exploitation search scheme which can search nearby for the global best position to refine the quality of the solution by using vibrated movement. It obtains a balance between exploration and exploitation by introducing a function of time ( $t$ ). RCS is modified from KMO and stochastically activated to reduce the computation cost and the loss of the population diversity. To evaluate the proposed algorithm, 10 benchmark datasets are tested, and the performance is compared with state-of-the-art works. Encouraging results are found in terms of efficiency and effectiveness of the proposed algorithm.

The remainder of this paper is organized as follows: Section 2 briefly describes the clustering problem and SSO. Section 3 introduces the proposed algorithm including VVS and RCS. Section 4 presents and discusses the computational results as well as the statistical analysis. Finally, conclusions are summarized in Section 5.

## Related works

### Clustering problem

The goal of clustering is to partition a given set of  $N$  objects  $Y = \{Y_1, Y_2, \dots, Y_N\}$ , each  $Y_i = \{y_1, y_2, \dots, y_D\} \in R^D$ , into  $K$  groups, also called clusters  $C = \{(C_1, Z_1), (C_2, Z_2), \dots, (C_K, Z_K)\}$ , where  $C_k$  represents the  $k$ th cluster,  $Z_k$  represents the centroid of the  $k$ th cluster, and  $K \leq N$ . The cluster structure is represented as follows [2]:

$$1. C_k \neq \phi, k = 1, 2, \dots, K; \tag{1}$$

$$2. \bigcup_{k=1}^K C_k = Y; \tag{2}$$

$$3. C_i \cap C_j = \phi; i, j = 1, 2, \dots, K \text{ and } i \neq j. \tag{3}$$

To split the objects into different clusters, many similarity criteria have been used in this task. One of most popular criterion is the Euclidean distance metric [3, 7, 9, 11, 16]. KM is an efficient and common clustering method which adopts this metric. The steps of this algorithm are as follows [17]:

Step 1: Randomly choose the  $K$  initial centroids of clusters,  $Z = \{Z_1, Z_2, \dots, Z_K\}$ .

Step 2: Measure the similarity by the Euclidean distance metric, and allocate object  $Y_i$  to  $C_k$ , if  $\|Y_i, Z_k\|^2 \leq \|Y_i, Z_p\|^2, p = 1, 2, \dots, K$ , and  $k \neq p$ .

Step 3: Evaluate the objective function named the sum of intra-cluster distances (SICD), as follows:

$$f(Y, C) = \sum_{k=1}^K \sum_{Y_i \in C_k} \|Y_i, Z_k\|^2 \tag{4}$$

Step 4: Recalculate new centroids of clusters, as follows:

$$Z_{new,k} = \frac{1}{N_k} \sum_{Y_i \in C_k} Y_i \tag{5}$$

Step 5: If  $Z_{new,k} = Z_k$ , then halt. Otherwise, continue from Step 2.

KM regards the clustering problem, which is an NP problem [18], as an optimization problem and aims at minimizing SICD, by assigning objects to the closest cluster centroid. KM is a well-known method in dealing with clustering problems because of its simplicity and efficiency. However, it suffers from the local optimum. This work introduces a novel SSO-based algorithm using the concept of KM to find optimal centroids by minimizing SICD without the initialization problem of KM.

### Simplified swarm optimization

SSO is a population-based algorithm proposed by Yeh in 2009 [19] to compensate for the deficiencies of PSO in solving discrete problems. This algorithm has recently been applied in many research areas because of its simplicity, efficiency, and flexibility [20–22].

In SSO, each individual in the swarm, called a particle representing a solution, is encoded as a finite-length string with a fitness value. Similar to many population-based algorithms, SSO improves the solution of a specified problem by the update mechanism (UM), which is the core of any evolutionary algorithm scheme. The UM of SSO is as follows:

$$x_{ij}^t = \begin{cases} x_{ij}^{t-1} & \text{if } \rho \in [0, C_w) \\ p_{ij}^{t-1} & \text{if } \rho \in [C_w, C_p) \\ g_j & \text{if } \rho \in [C_p, C_g) \\ x & \text{if } \rho \in [C_g, 1] \end{cases} \tag{6}$$

where  $x_{ij}^t$  is the position value in the  $i$ th particle with respect to the  $j$ th variable of the solution space at generation  $t$ .  $p_i = (p_{i1}, p_{i2}, \dots, p_{id})$ , where  $d$  is the total number of variables in the problem domain, represents the best solution with the best fitness value in its own history, known as  $pBest$ . The best solution with the best fitness value among all solutions is called  $gBest$ , which is denoted by  $g = (g_1, g_2, \dots, g_d)$ , and  $g_j$  denotes the  $j$ th variable in  $gBest$ .  $x$  is a new randomly

generated value between the lower bound and the upper bound of the  $j$ th variable.  $\rho$  is a uniform random number between 0 and 1.  $C_w$ ,  $C_p$ , and  $C_g$  are three predetermined parameters that form four interval probabilities. Thus,  $c_w = C_w$ ,  $c_p = C_p - C_w$ ,  $c_g = C_g - C_p$  and  $c_r = 1 - C_g$  represent the probabilities of the new variable from four sources, namely, the current solution,  $pBest$ ,  $gBest$ , and a random movement in the UM, respectively. The UM updates each particle to be a compromise of those four sources, particularly a random movement, which is different from the original PSO, maintains population diversity, and enhances the capacity of escaping from a local optimum.

## Proposed methods

The proposed algorithm is based on the original SSO, and combines with variable vibrating search (VVS) and rapid centralized strategy (RCS). This section introduces VVS, RCS and overall procedure for the proposed algorithm to solve the clustering problem.

### SSO clustering algorithm

Similar to many population-based algorithms, the SSO clustering algorithm randomly generates a population of particles, also called solutions. Encoding solution is the critical first step toward becoming a clustering algorithm. Assume that a clustering problem with  $D$  features is partitioned into  $K$  clusters, then  $Z = \{Z_1, Z_2, \dots, Z_K\}$  represents centroid vector and  $Z_k = \{z_{k1}, z_{k2}, \dots, z_{kD}\}$ , where  $k = 1, 2, \dots, K$ . Therefore, each solution string can be defined as  $X = \{x_1, x_2, \dots, x_{K \times D}\}$  as illustrated in Fig 1.

The primary steps of the SSO for clustering are summarized as follows:

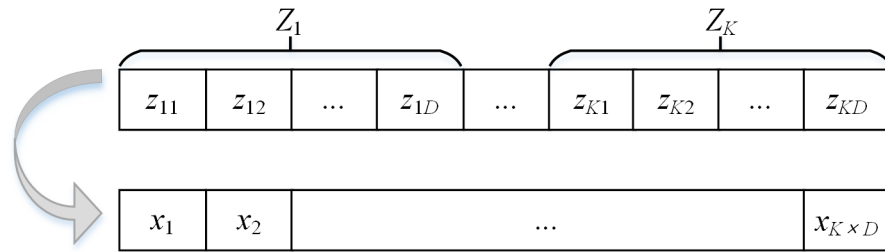
- Step 1: Generate a population of particles that represent centroids of each cluster with random positions based on a given dataset.
- Step 2: Evaluate the fitness value for each particle in the population according to Eq (4).
- Step 3: Update  $pBest$  and  $gBest$  if necessary.
- Step 4: Update the particle's variables according to Eq (6).
- Step 5: Stop the algorithm if the maximum number of iterations is met; otherwise, return to step 2.

### Variable vibrating search (VVS)

The UM promotes SSO to be an algorithm with the advantages of simplicity and flexibility. However, the UM is a stochastic process with insufficient capability to find nearby extreme points that can affect the efficiency and robustness of SSO, particularly in continuous problems. One way to improve the performance of SSO is adding the local search operation [23]. This modified SSO, called SSO-ELS, uses exchange local search scheme to find a new  $pBest$  of the particle or a new  $gBest$  by exchanging attributes with two randomly chosen particles in population. However, it consumes more time than the original SSO.

To overcome such exploitation problem of SSO, this work proposes an exploitation search scheme called VVS rather than a local search. It can refine the quality of the solution by searching the extreme points nearby the global best position. A new variable in a solution after VVS is calculated as Eq (7)

$$VVS(x_j) = g_j + (Ub_j - Lb_j) \times V(t) \tag{7}$$



**Fig 1. Example of a solution string.**

doi:10.1371/journal.pone.0137246.g001

where  $g_j$  denotes the  $j$ th variable in  $gBest$ .  $Lb_j$  and  $Ub_j$  are the lower and upper bounds of the  $j$ th variable, respectively. The amplitude constant  $V$  is a function of time ( $t$ ) as follow:

$$V(t) = \lambda \times \exp\left(-v \frac{iter}{Niter}\right) \tag{8}$$

where  $\lambda$  represents a random number uniformly distributed in  $[-1,1]$ .  $v$  is a predetermined parameter to handle the amplitude of variables.  $Niter$  is the total number of iterations and  $iter$  is the number of current iterations. As shown in Fig 2, with an increase in the number of iterations,  $V(t)$  moves in the plot as vibration wave and decreases toward zero at the last iteration. Hence, the particles search extreme points around the  $gBest$  position.

The balance between exploration and exploitation is an important criterion that can determine the performance of population-based optimization [24–25]. To tackle this problem, a function of time ( $t$ ),  $c_r(t)$  as Eq (9), is introduced to replace  $c_r$  which is a fixed constant in the original UM. This change leads the particles to stochastically explore the search spaces in beginning steps and gradually transform to exploit the extreme points nearby the  $gBest$  position as the number of iterations increases.

$$c_r(t) = c_r(0) \times \frac{iter}{Niter} \tag{9}$$

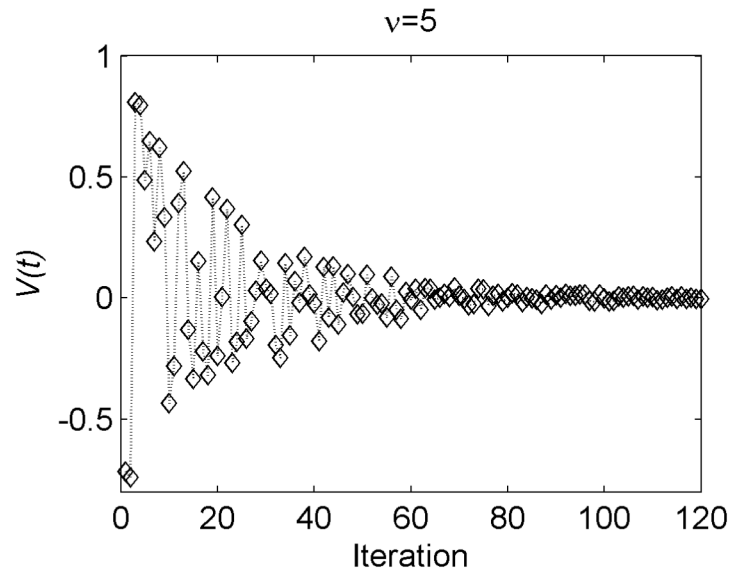
$$c_r(0) = 1 - C_g \tag{10}$$

However, the modified UM is more complex and consumes more time than the original UM because of the additional VVS. According to the result of preliminary tests as shown in S1 Table, the  $pBest$  scheme can be discarded in the modified UM to maintain the simplicity and efficiency of the proposed algorithm without affecting the performance of the proposed algorithm. Therefore, the UM of the proposed algorithm is modified as Eq (11)

$$x_{ij}^t = \begin{cases} x_{ij}^{t-1} & \text{if } \rho \in [0, C_w) \\ g_j & \text{if } \rho \in [C_w, C_g) \\ VVS(x_j) & \text{if } \rho \in [C_g, C_g + c_r(t)) \\ x & \text{if } \rho \in [C_g + c_r(t), 1] \end{cases} \tag{11}$$

### Rapid centralized strategy (RCS)

As mentioned in Section 1, the proposed algorithm may take more time to converge. Krishna and Murty [5] employed KMO to determine new cluster centers for all particles in each generation after initial population procedure by using Eq (5). However, overusing the arithmetic



**Fig 2. An amplitude of  $V(t)$  along with iteration when  $v = 5$ .**

doi:10.1371/journal.pone.0137246.g002

average may clamp the particles searching in the solution space because the members in clusters of different particles are the same after running a number of iterations. For example, if two particles in different positions have the same members in clusters, then both of them will move to exactly the same point, which is the mean point of the cluster after conducting KMO, as shown in Fig 3. In this case, particles may keep searching nearby the mean point rather than  $gBest$ . The computation cost is wasted, and the diversity of population is restricted by arithmetic average.

Some clustering algorithms take advantage of combining with KM and using its result as one of the particles for speeding up the convergence rate [9, 13, 16]. This strategy is named one-from-KM (OFK) in this work. However, these hybrid algorithms may start searching from a local optimum because of the initialization problem of the KM.

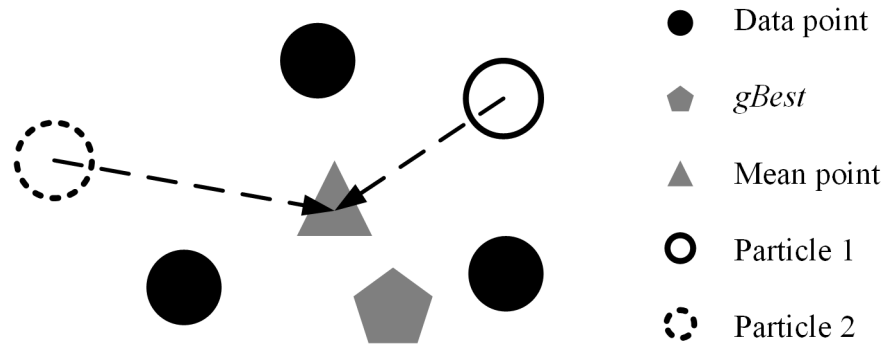
In this work, a modified accelerated strategy, RCS, is proposed. RCS is inspired by KMO and can efficiently find a better centroid of a cluster at the initial state and keep the diversity of population. The following two steps constitute RCS:

Step 1: In the initial state, the cluster centroids of all particles are recalculated according to Eq (5), after being generated randomly by the proposed algorithm.

Step 2: In each iteration, only some particles are recalculated by RCS depending on its random number.

Recalculating the centroids of all particles in the initial state can escape from the initialization problem of the KM through the diversity of particles and then obtain a promising initial solution. In step 2, a random number,  $rand$ , belonging to  $[0, 1]$  is generated for each particle after the particle is updated by the modified UM. If  $rand < \beta$ , then the RCS is implemented to recalculate the cluster centroids of the particle.  $\beta$  is a predetermined parameter in the interval  $[0, 1]$  to decide the proportion of particles that need to be recalculated the centroids.

Based on the above description, the proposed algorithm is called VSSO-RCS, and its steps are shown in Fig 4.



**Fig 3. Particles moving behavior in accelerated strategy.**

doi:10.1371/journal.pone.0137246.g003

## Experiment Results and Discussion

The experiments are composed of two parts. In the first experiment, RCS is compared with other accelerated strategies, including KMO and OFK, to illustrate the effectiveness of the proposed accelerated strategy. In the second experiment, well-known and recently developed population-based algorithms are implemented to evaluate the performance of the proposed algorithm. These population-based algorithms include SSO [17], SSO-ELS [23], BH [3], KSRPSO [13], and GSA-KM [16]. All experiments are executed in MATLAB R2012b on a computer equipped with an Intel 2.4 GHz CPU and 12 GB of memory.

As mentioned in Section 2, this work uses SICD as a criterion to evaluate the performance of all clustering algorithms. SICD is the sum of the distances between objects in the same cluster, as defined in Eq (4). A smaller SICD results in a higher quality solution.

### Data sets

According to the categorization of dataset size described by Kudo and Sklansky [26], the problem can be categorized into three categories in terms of the number of features: small with  $0 < D \leq 19$ , medium with  $20 < D \leq 49$  and large with  $D \geq 50$ . Twelve datasets, which are taken from the UCI repository database [27], cover all categories to test all approaches implemented in this work. The characteristics of these datasets are summarized in Table 1. These datasets have sizes ranging from hundreds to thousands and the feature size ranges from 3 to 60. All datasets contain numeric features with no missing data.

### Parameter settings

The algorithmic parameters for all approaches are illustrated in Table 2. Parameter settings may influence the quality of results and the settings of each approach are suggested from previous studies. In KSRPSO, the settings of cognition  $c_1$  and social  $c_2$  parameters are 0.5 and 2.5, respectively. The inertia weight  $w$  is equal to  $0.5 \times rand / 2$ , where  $rand$  is a uniformly generated random number between 0 and 1. Three parameters,  $f$ ,  $d$  and  $a$ , are set to 0.2, 0.2 and 0.8 respectively, controlling the selective particle regeneration mechanism for local search [13]. The parameters in SSO, SSO-ELS and SSO-RCS,  $C_w$ ,  $C_p$  and  $C_g$ , are all set at 0.1, 0.4 and 0.9, respectively [17, 23]. In VSSO-RCS,  $C_w$ ,  $C_g$ ,  $\nu$  and  $\beta$  are set at 0.2, 0.9, 10 and 0.1, respectively. Based on the preliminary test this parameter setting of VSSO-RCS provides a good chance of finding the global optimal solution. Two parameters in GSA-KM, the initial gravitational constant  $G_0$  and  $\alpha$ , are set at 100 and 20, respectively [16, 28]. BH has no parameters need to be set [3].



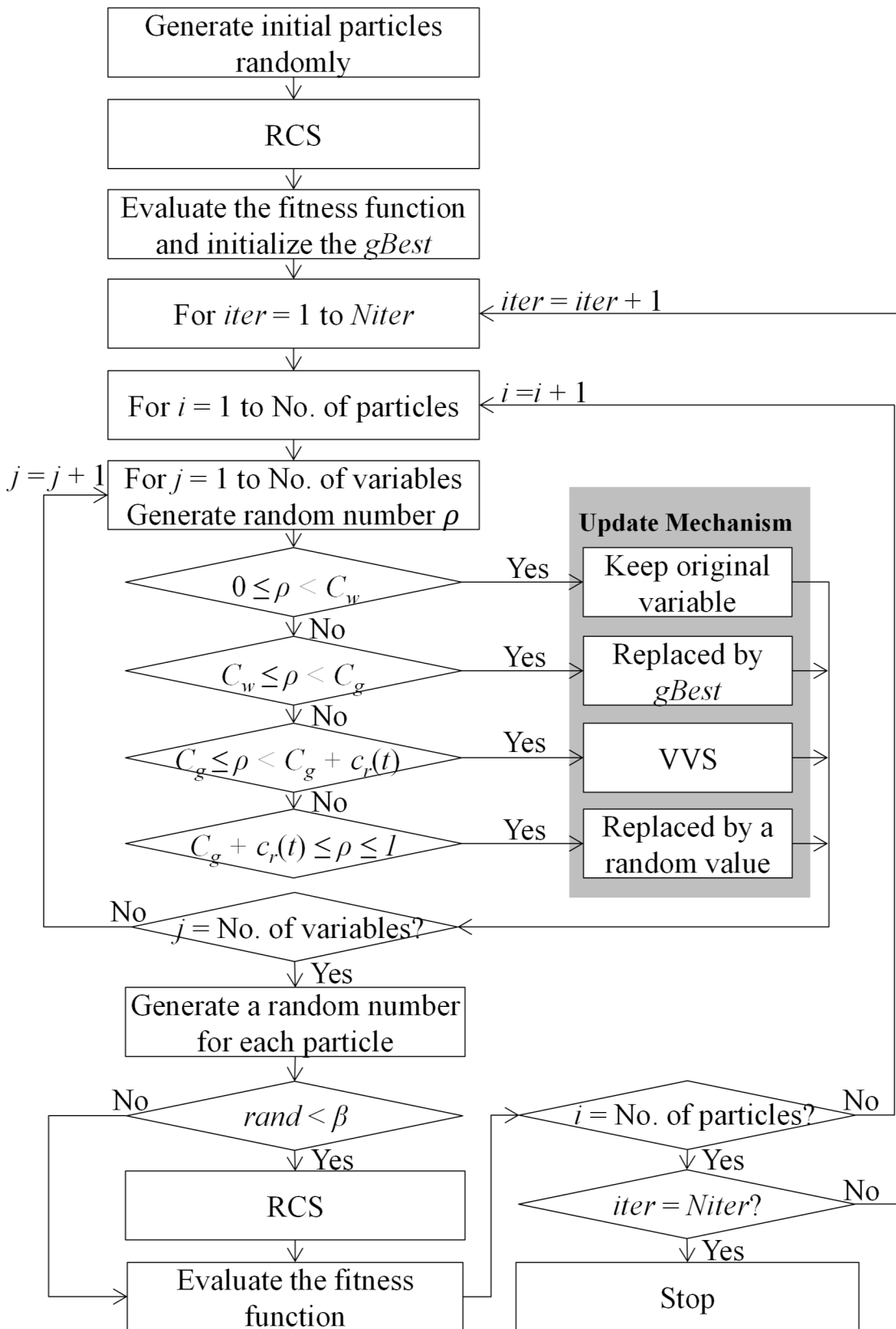


Fig 4. Flowchart of the VSSO-RCS algorithm.

doi:10.1371/journal.pone.0137246.g004



**Table 1. Characteristics of the considered data sets.**

Categorization	Dataset	Features ( <i>D</i> )	Instances ( <i>N</i> )	Clusters ( <i>K</i> )
Small	Vowel	3	871	6
	Iris	4	150	3
	Crude oil	5	56	3
	Cancer	9	683	2
	CMC	9	1473	3
	Glass	9	214	6
	MG Telescope	10	19020	2
	Wine	13	178	3
	EGG eye	15	14980	2
Medium	WDBC	30	569	2
	Ionosphere	34	351	2
Large	Sonar	60	208	2

doi:10.1371/journal.pone.0137246.t001

For each run, the number of iterations and population size of each approach depends on the number of features and clusters of each dataset.  $10 \times \Psi$  iterations and  $3 \times \Psi$  population size are performed, and  $\Psi$  is equal to  $D \times K$ . [11, 12].

### Results and discussion

**Experiment 1: Evaluation of RCS.** KM, SSO, SSO-KMO, SSO-OFK and SSO-RCS are implemented to demonstrate the effectiveness of the proposed RCS. SSO-KMO, SSO-OFK, and SSO-RCS represent SSO that adopted the accelerated strategies KMO, OFK, and RCS, respectively. Four datasets are selected among three categories of datasets for this experiment, including Cancer, Glass, Ionosphere, and Sonar. The experimental results summarized in Table 3 include SICD and CPU time (CT). SICD is given in terms of the best, average, worst, and standard deviation (Std.) of the obtained solutions after 30 runs. CPU time is a record of the average proceeding time of 30 runs.

As shown in Table 3, where the best average values are shown in bold, SSO without any accelerated strategy produces the worst solution on all selected datasets in comparison with SSO-KMO, SSO-OFK, and SSO-RCS. This result confirms that the three accelerated strategies are functional and facilitate SSO to find better solutions. SSO-KMO consumes more time and obtains worse solution than SSO-RCS. This result proves that RCS modified from KMO not only reduces the computational cost but also enhances the performance of SSO. As expected,

**Table 2. Value of parameters in six algorithms.**

Parameter <sup>a</sup>	KSRPSO	SSO-based	VSSO-RCS	GSA-KM
Value				
$c_1, c_2$	0.5, 2.5	-	-	-
$w$	$0.5 \times rand / 2$	-	-	-
$f, d, a$	0.2, 0.2, 0.8	-	-	-
$C_w, C_p, C_g$	-	0.1, 0.4, 0.9	0.2, -, 0.9	-
$v, \beta$	-	-	10, 0.1	-
$G_0, \alpha$	-	-	-	100, 20

<sup>a</sup> There is no parameter need to be set in BH [3].

doi:10.1371/journal.pone.0137246.t002

SSO-OFK is faster than the other strategies. However, the solutions obtained by SSO-OFK are all worse than those obtained by SSO-RCS. The results reveal that SSO-RCS yields higher quality solutions on all selected datasets than SSO-KMO and SSO-OFK. Obviously, RCS outperforms other strategies in this experiment.

Fig 5 depicts the progress of the average *gBest* over 30 runs, providing insights into the convergence behavior of SSO-KMO, SSO-OFK, and SSO-RCS. For all datasets, SSO exhibits the worst convergence pattern. SSO-KMO cannot obtain any benefit of yielding initial solutions. Furthermore, it displays a fast but premature convergence to a local optimum on Glass, Ionosphere and Sonar dataset, as shown in Fig 5(B), 5(C) and 5(D). This result is understandable because the global search capability depends on the diversity of population. KMO recalculates the centroids of all particles at each iteration using the arithmetic average. After a number of iterations, the overuse of the arithmetic average may backfire and compel the particle to be the same as its predecessor or neighbors. As a result, KMO may be disabled, and the population diversity would be diminished.

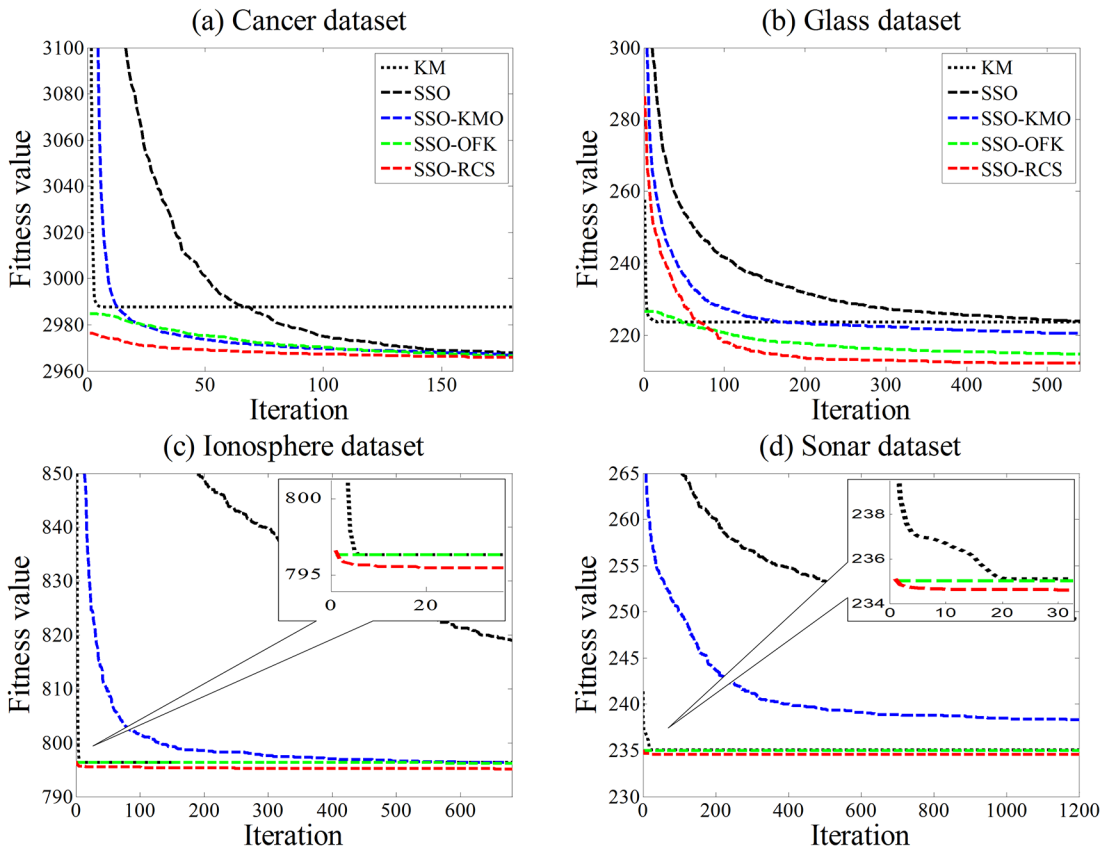
OFK using the result of KM as one of the initial particles can offer SSO-OFK promising initial solutions that lead to initially converging faster than SSO-KMO. However, the initial solutions obtained from KM may suffer from local optimum. Therefore, *gBest* and *pBest* are initialized from a pitfall and seriously affect the capability of global search, as shown in Fig 5(B), 5(C) and 5(D).

SSO-RCS demonstrates a consistent performance pattern among all the considered accelerated strategies. RCS not only produces promising initial solutions but also aids SSO to find higher quality solutions than SSO-KMO and SSO-OFK on all of the selected datasets. Results show that the proposed RCS is an ideal choice for accelerating convergence speed in clustering algorithms.

**Table 3. Results of three accelerated strategies.**

Dataset	Criteria	KM	SSO	SSO-KMO	SSO-OFK	SSO-RCS
Cancer	Best	2,986.96	2,965.13	2,965.56	2,964.82	2,964.93
	Avg.	2,987.84	2,967.87	2,967.11	2,966.65	<b>2,965.91</b>
	Worst	2,988.43	2,975.96	2,969.20	2,970.40	2,967.93
	Std.	0.73	2.55	1.05	1.40	0.82
	CT(s)	1.45	5.52	10.84	5.79	5.83
Glass	Best	213.24	213.52	211.53	211.70	211.34
	Avg.	223.58	223.85	220.55	214.71	<b>212.11</b>
	Worst	253.83	240.66	243.43	218.57	215.60
	Std.	10.21	9.74	10.00	1.65	1.01
	CT(s)	2.75	45.64	85.25	48.05	48.42
INSP	Best	796.33	811.75	795.90	795.42	795.13
	Avg.	796.40	817.62	796.23	796.10	<b>795.37</b>
	Worst	796.47	824.36	796.66	796.33	795.59
	Std.	0.07	3.72	0.28	0.28	0.15
	CT(s)	1.88	110.06	197.01	108.65	114.02
Sonar	Best	234.77	235.25	240.00	234.72	234.51
	Avg.	235.10	238.32	240.42	235.06	<b>234.58</b>
	Worst	235.21	240.32	241.66	235.21	234.60
	Std.	0.15	2.33	0.50	0.20	0.03
	CT(s)	2.29	664.94	693.93	432.56	425.46

doi:10.1371/journal.pone.0137246.t003



**Fig 5. Convergence comparison of different accelerated strategies.**

doi:10.1371/journal.pone.0137246.g005

**Experiment 2: Evaluation of VSSO-RCS.** Well-known and recent population-based algorithms including SSO, SSO-RCS, SSO-ELS, BH, KSRPSO, and GSA-KM are implemented to demonstrate the effectiveness of the proposed algorithm, VSSO-RCS. Experimental results tested on 12 benchmark datasets are summarized in Tables 4–6, where the best values are shown in bold and all results are obtained after 30 runs.

As illustrated in Tables 4–6, VSSO-RCS produces better results than SSO-RCS on all of the datasets. This result proves that the proposed exploitation scheme VVS plays its prescribed roles in exploitation search and can refine the solutions. For the Vowel, Crude oil, Cancer, CMC, Glass and EGG eye datasets, the average values obtained by VSSO-RCS are 149148.08, 277.26, 2964.39, 5532.18, 211.31 and 2354849.12, respectively, which are better than those obtained by the other algorithms. For the Cancer, CMC, EGG eye and WDBC datasets, the worst values obtained by VSSO-RCS are 2964.39, 5532.18, 2355129.21 and 149473.86, respectively, which are better than the best values yielded by the other test algorithms except for BH and GSA-KM. This result means that other SSO-based algorithms and KSRPSO are powerless to achieve those values even once within 30 runs. Moreover, the standard deviation on the Iris, Cancer, CMC, WDBC, Ionosphere, and Sonar datasets are 0.00, which means that the proposed algorithm is more reliable than the other algorithms.

In terms of CPU time, the proposed algorithm is faster than other SSO-based algorithms. This advantage is attributed to the modified UM of VSSO-RCS by discarding the *pBest* scheme. GSA-KM is time consuming because of its computational complexity. In general, BH and

**Table 4. Results obtained by the algorithms on Vowel, Iris, Crude oil and Cancer datasets.**

Dataset	Criteria	SSO	SSO-RCS	SSO-ELS	BH	KSRPSO	GSA-KM	VSSO-RCS
Vowel	Best	149,247.64	149,021.16	149,096.69	<b>148,967.24</b>	149,089.96	149,076.71	<b>148,967.24</b>
	Avg	150,421.69	149,315.13	150,118.24	151,684.62	151,758.39	152,289.92	<b>149,148.08</b>
	Worst	154,119.51	150,267.43	153,232.13	168,379.82	170,433.64	158,612.03	<b>150,139.66</b>
	Std	1,249.07	394.08	863.56	5,100.30	4,205.04	2,947.95	<b>336.16</b>
	CT	6.54	6.97	10.86	<b>6.13</b>	6.65	13.38	6.44
Iris	Best	96.75	96.67	96.72	<b>96.66</b>	96.68	<b>96.66</b>	<b>96.66</b>
	Avg	97.11	96.73	97.17	<b>96.66</b>	98.98	96.71	<b>96.66</b>
	Worst	98.06	96.82	97.94	96.71	127.67	97.22	<b>96.66</b>
	Std	0.35	0.04	0.42	0.01	6.96	0.13	<b>0.00</b>
	CT	0.70	0.74	1.20	<b>0.61</b>	0.75	1.85	<b>0.61</b>
Crude oil	Best	277.25	277.25	277.24	<b>277.21</b>	277.22	277.21	<b>277.21</b>
	Avg	277.53	277.35	277.99	277.27	277.35	277.63	<b>277.26</b>
	Worst	278.21	277.42	293.58	<b>277.30</b>	277.86	285.76	277.36
	Std	0.25	<b>0.04</b>	2.95	<b>0.04</b>	0.13	1.56	0.05
	CT	0.99	1.06	1.64	<b>0.83</b>	1.02	3.76	<b>0.83</b>
Cancer	Best	2,965.13	2,964.93	2,964.95	2,964.39	2,964.86	2,965.00	<b>2,964.39</b>
	Avg	2,967.87	2,965.91	2,967.05	2,964.40	2,966.32	2,973.42	<b>2,964.39</b>
	Worst	2,975.96	2,967.93	2,970.83	2,964.41	2,969.62	2,985.84	<b>2,964.39</b>
	Std	2.55	0.82	1.32	0.00	1.24	6.57	<b>0.00</b>
	CT	5.52	5.83	10.03	<b>5.16</b>	5.90	11.28	5.39

doi:10.1371/journal.pone.0137246.t004

VSSO-RCS take less processing time than KSRPSO on small and medium size datasets. KSRPSO has superiority on large size dataset.

**The statistical analysis.** A nonparametric statistical analysis, the Friedman test, is conducted to confirm whether the proposed VSSO-RCS offers a significant improvement. If statistically significant differences exist among all algorithms, then the Holm’s method is employed as a post hoc test to compare the proposed algorithm (control algorithm) and the other algorithms. The significance level is set to  $\alpha = 0.05$  to determine whether or not the hypothesis is rejected in all cases. These tests are detailed in [29].

Table 7 reports the average ranks computed through the Friedman test based on the average values of SICD. The table shows that the average rank of the proposed VSSO-RCS is the smallest among the algorithms. Therefore, the proposed method is the best performing one, followed by BH, SSO-RCS, GSA-KM, KSRPSO, SSO-ELS, and SSO, successively.

The *p*-Values computed through the Friedman test is given in Table 8, suggesting significant differences in the average values of SICD among the considered approaches.

To determine sufficient statistical differences between VSSO-RCS and the remaining algorithms, the Holm’s method is conducted as a post hoc test.

Table 9 shows that all *p*-values are smaller than  $\alpha = 0.05$ , which indicates that the control algorithm VSSO-RCS is statistically better than SSO, SSO-RCS, SSO-ELS, BH, KSRPSO, and GSA-KM in term of SICD.

The same procedure is conducted to check whether or not significant differences in CPU time exist between the clustering algorithms. The results are shown in Tables 10–12. The Friedman’s test reveals that the proposed algorithm is ranked second behind BH and that statistically significant differences in the average CPU time exist among the algorithms. Table 12 indicates that VSSO-RCS is more efficient than SSO-RCS, SSO-ELS, and GSA-KM. No significant difference is found among SSO, KSRPSO, BH, and VSSO-RCS.

**Table 5. Results obtained by the algorithms on CMC, Glass, MG Telescope and Wine datasets.**

Dataset	Criteria	SSO	SSO-RCS	SSO-ELS	BH	KSRPSO	GSA-KM	VSSO-RCS
CMC	Best	5,533.46	5,532.77	5,532.76	5,532.19	5,532.36	5,532.19	<b>5,532.18</b>
	Avg	5,535.93	5,533.45	5,534.84	5,532.24	5,532.87	5,532.19	<b>5,532.18</b>
	Worst	5,540.84	5,534.59	5,539.57	5,532.45	5,533.59	5,532.19	<b>5,532.18</b>
	Std	1.77	0.45	1.48	0.05	0.32	<b>0.00</b>	<b>0.00</b>
	CT	27.90	28.66	46.22	<b>26.10</b>	27.45	54.62	27.12
Glass	Best	213.52	211.34	211.19	237.65	<b>210.36</b>	213.11	210.43
	Avg	223.85	212.11	218.38	260.93	217.87	218.52	<b>211.31</b>
	Worst	240.66	215.60	235.35	269.64	245.32	227.58	<b>214.81</b>
	Std	9.74	<b>1.01</b>	8.50	8.36	7.91	3.60	1.78
	CT	45.64	48.42	78.82	38.66	<b>36.94</b>	508.16	39.26
MG Telescope	Best	1,623,698.99	1,623,296.10	1,623,772.73	1,623,042.28	1,623,322.11	<b>1,623,042.27</b>	1,623,042.28
	Avg	1,631,490.90	1,625,505.00	1,629,648.90	1,623,042.31	1,627,770.46	<b>1,623,042.27</b>	1,623,045.45
	Worst	1,637,591.98	1,632,364.00	1,635,628.40	1,623,042.38	1,635,781.99	<b>1,62,3042.27</b>	1,623,072.86
	Std	4,685.46	3,544.38	3,877.04	0.03	4,191.48	<b>0.00</b>	9.63
	CT	1145.16	1175.22	1709.08	<b>988.84</b>	1211.86	1341.66	1084.09
Wine	Best	16,292.44	16,292.45	16,292.43	16,292.19	16,292.22	16,292.67	<b>16,292.18</b>
	Avg	16,294.07	16,293.86	16,294.14	<b>16,292.71</b>	16,292.78	16,293.90	16,292.76
	Worst	16,296.90	16,295.72	16,295.67	<b>16,294.17</b>	16,294.47	<b>16,294.17</b>	<b>16,294.17</b>
	Std	1.09	0.89	1.05	0.70	0.67	<b>0.45</b>	0.82
	CT	20.00	21.65	34.01	<b>17.64</b>	18.33	141.91	18.08

doi:10.1371/journal.pone.0137246.t005

**Table 6. Results obtained by the algorithms on EGG eye, WDBC, INSP and Sonar datasets.**

Dataset	Criteria	SSO	SSO-RCS	SSO-ELS	BH	KSRPSO	GSA-KM	VSSO-RCS
EGG eye	Best	8,032,669.11	5,644,264.31	2,385,500.94	<b>2,354,713.85</b>	3,010,467.48	2,778,514.06	2,354,756.19
	Avg	16,361,259.24	14,464,277.41	2,505,596.68	2,586,299.09	3,210,719.24	2,791,675.62	<b>2,354,849.12</b>
	Worst	26,910,999.74	29,333,498.64	2,762,809.53	3,214,000.36	3,456,696.67	2,867,748.16	<b>2,355,129.21</b>
	Std	6,668,788.32	6,757,831.04	108,461.85	271,990.85	114,894.72	27,242.75	<b>129.73</b>
	CT	856.77	874.94	1116.37	850.98	880.40	912.04	<b>849.99</b>
WDBC	Best	149,474.46	149,474.20	149,474.40	<b>149,473.86</b>	149,473.89	<b>149,473.86</b>	<b>149,473.86</b>
	Avg	149,480.77	149,477.49	149,479.23	<b>149,473.86</b>	149,474.13	<b>149,473.86</b>	<b>149,473.86</b>
	Worst	149,494.82	149,483.18	149,493.45	149,473.87	149,474.62	<b>149,473.86</b>	<b>149,473.86</b>
	Std	5.93	2.76	4.10	<b>0.00</b>	0.20	<b>0.00</b>	<b>0.00</b>
	CT	94.57	105.77	185.15	90.91	<b>90.58</b>	732.59	90.76
INSP	Best	814.52	794.96	794.32	793.92	793.78	<b>793.71</b>	<b>793.71</b>
	Avg	819.07	795.13	794.86	794.30	793.87	<b>793.71</b>	<b>793.71</b>
	Worst	827.72	795.30	796.37	795.34	794.02	<b>793.71</b>	793.72
	Std	4.76	0.10	0.61	0.42	0.07	<b>0.00</b>	<b>0.00</b>
	CT	105.74	114.02	195.46	105.38	<b>95.97</b>	1,171.57	96.35
Sonar	Best	247.73	234.51	238.85	234.22	233.77	<b>233.76</b>	<b>233.76</b>
	Avg	249.19	234.58	239.87	245.02	233.86	<b>233.76</b>	<b>233.76</b>
	Worst	251.12	234.60	245.27	266.59	234.08	<b>233.76</b>	233.77
	Std	1.11	0.03	1.93	14.97	0.09	<b>0.00</b>	<b>0.00</b>
	CT	395.85	425.46	650.07	347.83	<b>301.61</b>	11,132.94	328.31

doi:10.1371/journal.pone.0137246.t006

**Table 7. Results of Friedman ranks on the average of SICD.**

Dataset	SSO	SSO-RCS	SSO-ELS	BH	KSRPSO	GSA-KM	VSSO-RCS
Vowel	4.00	2.00	3.00	5.00	6.00	7.00	1.00
Iris	5.00	4.00	6.00	1.50	7.00	3.00	1.50
Crude oil	5.00	3.50	7.00	2.00	3.50	6.00	1.00
Cancer	6.00	3.00	5.00	2.00	4.00	7.00	1.00
CMC	7.00	5.00	6.00	3.00	4.00	2.00	1.00
Glass	6.00	2.00	4.00	7.00	3.00	5.00	1.00
MG T	7.00	4.00	6.00	2.00	5.00	1.00	3.00
Wine	6.00	4.00	7.00	1.00	3.00	5.00	2.00
EGG eye	7.00	6.00	2.00	3.00	5.00	4.00	1.00
WDBC	7.00	5.00	6.00	2.00	4.00	2.00	2.00
Ionosphere	7.00	6.00	5.00	4.00	3.00	1.50	1.50
Sonar	7.00	4.00	5.00	6.00	3.00	1.50	1.50
Average	6.17	4.04	5.17	3.21	4.21	3.75	1.46

doi:10.1371/journal.pone.0137246.t007

**Table 8. Results of Friedman tests on the average of SICD.**

Method	Statistical value	p-Value	Hypothesis
Friedman	34.07	0.000	Rejected

doi:10.1371/journal.pone.0137246.t008

**Table 9. Results of the post hoc test on the average of SICD.**

Method	z-Value	p-Value	Hypothesis
SSO	5.339	0.000	Rejected
VSSO-RCS	2.929	0.003	Rejected
SSO-ELS	4.205	0.000	Rejected
BH	1.984	0.047	Rejected
KSRPSO	3.118	0.002	Rejected
GSA-KM	2.598	0.009	Rejected

doi:10.1371/journal.pone.0137246.t009

The results of empirical and statistical analyses show that VSSO-RCS is not that superior in CPU time compared with SSO, BH and KSRPSO. However, VSSO-RCS exhibits promising and effective clustering performance.

**Empirical analysis of algorithm efficiency.** The results from the previous subsection demonstrate that the proposed algorithm can perform better than its competitors in terms of the quality of solutions. Also, it can be observed that the problem size may impact the computation time. To better observe the effect of the number of data instances and features on the proposed method, two groups of artificial datasets are generated. As shown in Table 13, the first group varies the number of instances from 16000 to 2048000 at the fixed number of features (i.e.  $D = 6$ ) for evaluating the effect of instance size. The second group varies the number of features from 400 to 51200 at the fixed number of instances ( $N = 1000$ ) for observing the effect of feature size as shown in Table 14. The number of clusters in both of the two groups is fixed to  $K = 2$ . The proposed method performs 10 independent runs on each dataset with 100 iterations and 30 population size.

**Table 10. Results of Friedman ranks on the average of CPU time.**

Dataset	SSO	SSO-RCS	SSO-ELS	BH	KSRPSO	GSA-KM	VSSO-RCS
Vowel	3.00	5.00	6.00	1.00	4.00	7.00	2.00
Iris	3.00	4.00	6.00	1.50	5.00	7.00	1.50
Crude oil	3.00	5.00	6.00	1.50	4.00	7.00	1.50
Cancer	3.00	4.00	6.00	1.00	5.00	7.00	2.00
CMC	4.00	5.00	6.00	1.00	3.00	7.00	2.00
Glass	4.00	5.00	6.00	2.00	1.00	7.00	3.00
MG T	3.00	4.00	7.00	1.00	5.00	6.00	2.00
Wine	4.00	5.00	6.00	1.00	3.00	7.00	2.00
EGG eye	3.00	4.00	7.00	2.00	5.00	6.00	1.00
WDBC	4.00	5.00	6.00	3.00	1.00	7.00	2.00
INSP	4.00	5.00	6.00	3.00	1.00	7.00	2.00
Sonar	4.00	5.00	6.00	3.00	1.00	7.00	2.00
Average	3.50	4.67	6.17	1.75	3.17	6.83	1.92

doi:10.1371/journal.pone.0137246.t010

**Table 11. Results of Friedman tests on the average of CPU time.**

Method	Statistical value	p-Value	Hypothesis
Friedman	60.464	0.000	Rejected

doi:10.1371/journal.pone.0137246.t011

**Table 12. Results of the post hoc test on the average of SICD.**

Method	z-Value	p-Value	Hypothesis
SSO	1.795	0.073	Not rejected
VSSO-RCS	3.118	0.002	Rejected
SSO-ELS	4.819	0.000	Rejected
BH	0.189	1.150	Not rejected
KSRPSO	1.417	0.156	Not rejected
GSA-KM	5.575	0.000	Rejected

doi:10.1371/journal.pone.0137246.t012

**Table 13. Empirical analysis result of varying in number of data instances.**

<i>i</i>	1	2	3	4	5	6	7	8
<b><i>N</i> (<i>D</i> = 6)</b>	<b>16000</b>	<b>32000</b>	<b>64000</b>	<b>128000</b>	<b>256000</b>	<b>512000</b>	<b>1024000</b>	<b>2048000</b>
Max CT	15.21	29.72	56.83	121.20	246.08	496.77	999.67	2149.43
Mean CT	14.47	28.49	55.80	119.06	241.86	484.74	966.41	1958.85
Min CT	13.85	27.38	54.90	116.95	237.07	473.70	945.32	1989.48
Std CT	0.48	0.73	0.69	1.26	2.77	6.65	18.57	51.56
Ratio <sup>a</sup>		1.97	1.96	2.13	2.03	2.00	1.99	2.03

<sup>a</sup>Ratio = Mean CT<sub>*i*</sub> / Mean CT<sub>*i-1*</sub>

doi:10.1371/journal.pone.0137246.t013

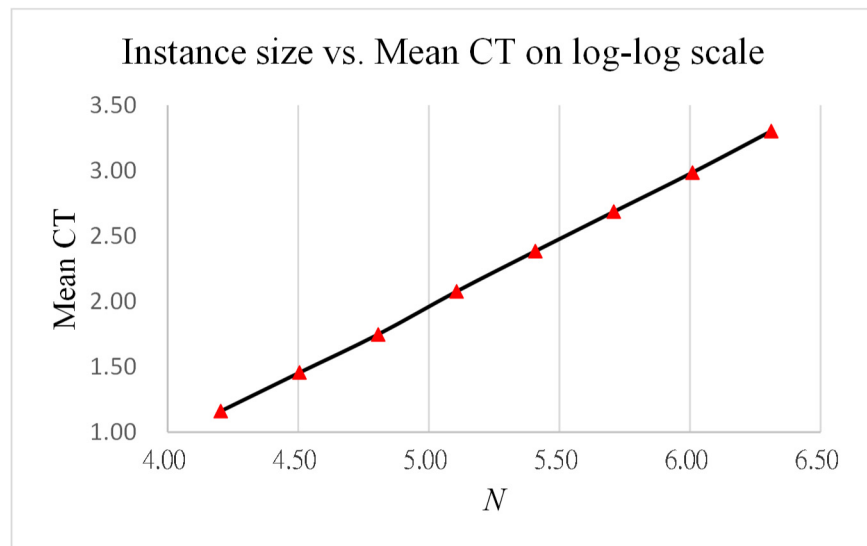
As reported in Tables 13 and 14, the results obtained from the two analysis cases indicate that the ratios of CPU time increase both converge to 2. Fig 6 and Fig 7 show log-log plots of



**Table 14. Empirical analysis result of varying in number of data features.**

<i>i</i>	1	2	3	4	5	6	7	8
<i>D</i> ( <i>N</i> = 1000)	400	800	1600	3200	6400	12800	25600	51200
Max CT	55.82	109.48	217.14	434.14	881.76	1764.43	3592.34	7413.43
Mean CT	54.76	108.70	214.54	431.66	877.09	1754.31	3574.15	7227.01
Min CT	53.43	107.58	211.54	428.05	871.33	1727.27	3557.24	7230.68
Std CT	0.71	0.60	1.95	2.07	2.88	10.68	10.60	54.29
Ratio		1.99	1.97	2.01	2.03	2.00	2.04	2.02

doi:10.1371/journal.pone.0137246.t014



**Fig 6. Log-log plot of CPU time vs. instance size.**

doi:10.1371/journal.pone.0137246.g006

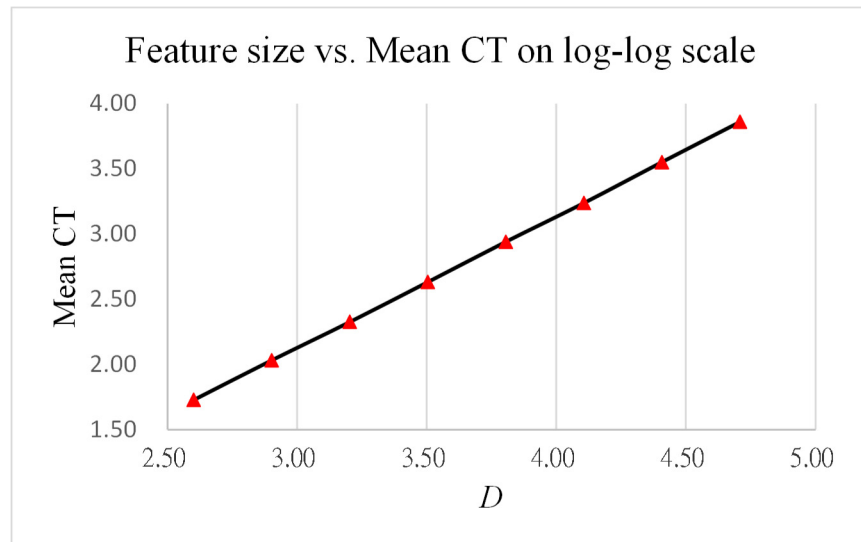
CPU time vs. instance size and vs. feature size, respectively, both indicating that the computation time grows with the size of instance and feature in a linear and stable manner.

## Conclusions

The proposed VSSO-RCS is a modified version of simplified swarm optimization to effectively and efficiently tackle the clustering problems. VSSO overcomes the exploitation problem in SSO and facilitates VSSO-RCS to refine the quality of solution. VSSO-RCS also converges fast to an optimum solution by adopting RCS.

To assess the performance of VSSO-RCS, two experiments are conducted. First, RCS is compared with other two powerful accelerated strategies. RCS not only can obtain promising initial solutions but also converge more efficiently than KMO and OFK. Second, VSSO-RCS is compared with state-of-the-art population-based algorithms on 12 benchmark datasets. Results reveal that VSSO-RCS is superior to its competitors from the perspective of solution quality and is efficient in terms of the processing time required.

Our results on empirical analysis suggest that instance and feature size both affect the computation efficiency of the proposed algorithm. As the problem size grows, it adds more challenges to the computation time of our proposed method, especially for dealing with very



**Fig 7. Log-log plot of CPU time vs. feature size.**

doi:10.1371/journal.pone.0137246.g007

large-size problems (i.e. big data) [30]. Due to such aspect, our future research will focus on the following:

1. Apply effective techniques [31–35] which can reduce the search space into the proposed algorithm to mitigate such an issue.
2. Modify the proposed method in a parallel or distributed form [36–38] to improve the computation efficiency.

In addition, it is also worth exploring other potential application, such as classification, fully utilize the VSSO-RCS.

## Supporting Information

**S1 Table. Results of preliminary tests.**

(DOCX)

## Author Contributions

Conceived and designed the experiments: WCY CML. Performed the experiments: CML. Analyzed the data: CML. Contributed reagents/materials/analysis tools: WCY. Wrote the paper: CML.

## References

1. Chaves AA, Lorena LAN. Clustering search algorithm for the capacitated centered clustering problem. *Computers & Operations Research*. 2010; 37(3): 552–558.
2. Xu R, Wunsch D. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*. 2005; 16(3): 645–678. PMID: [15940994](#)
3. Hatamlou A. Black hole: A new heuristic optimization approach for data clustering. *Information Sciences*. 2013; 222: 175–184.
4. Selim SZ, Ismail MA. K-means-type algorithms: a generalized convergence theorem and characterization of local optimality. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 1984; 6(1): 81–87. PMID: [21869168](#)

5. Krishna K, Murty MN. Genetic K-means algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*. 1999; 29(3): 433–439.
6. Maulik U, Bandyopadhyay S. Genetic algorithm-based clustering technique. *Pattern recognition*. 2000; 33(9): 1455–1465.
7. Liu Y, Yi Z, Wu H, Ye M, Chen K. A tabu search approach for the minimum sum-of-squares clustering problem. *Information Sciences*. 2008; 178(12): 2680–2704.
8. Shelokar PS, Jayaraman VK, Kulkarni BD. An ant colony approach for clustering. *Analytica Chimica Acta*. 2004; 509(2): 187–195.
9. Zhang L, Cao Q. A novel ant-based clustering algorithm using the kernel method. *Information Sciences*. 2011; 181(20): 4658–4672.
10. Karaboga D, Ozturk C. A novel clustering approach: Artificial Bee Colony (ABC) algorithm. *Applied Soft Computing*. 2011; 11(1): 652–657.
11. Kao YT, Zahara E, Kao IW. A hybridized approach to data clustering. *Expert Systems with Applications*. 2008; 34(3): 1754–1762.
12. Chuang LY, Hsiao CJ, Yang CH. Chaotic particle swarm optimization for data clustering. *Expert systems with Applications*. 2011; 38(12): 14555–14563.
13. Tsai CY, Kao IW. Particle swarm optimization with selective particle regeneration for data clustering. *Expert Systems with Applications*. 2011; 38(6): 6565–6576.
14. Selim SZ, Alsultan K. A simulated annealing algorithm for the clustering problem. *Pattern recognition*. 1991; 24(10): 1003–1008.
15. Hatamlou A, Abdullah S, Hatamlou M. Data clustering using big bang–big crunch algorithm. In *Innovative Computing Technology*. 2011; 241: 383–388.
16. Hatamlou A, Abdullah S, Nezamabadi-pour H. A combined approach for clustering based on K-means and gravitational search algorithms. *Swarm and Evolutionary Computation*. 2012; 6: 47–52.
17. Bandyopadhyay S, Maulik U. An evolutionary technique based on K-means algorithm for optimal clustering in  $R^N$ . *Information Sciences* 2002; 146(1): 221–237.
18. Adib AB. NP-hardness of the cluster minimization problem revisited. *Journal of Physics A: Mathematical and General*. 2005; 38(40): 8487–8494.
19. Yeh WC, Chang WW, Chung YY. A new hybrid approach for mining breast cancer pattern using discrete particle swarm optimization and statistical method. *Expert Systems with Applications*. 2009; 36(4): 8204–8211.
20. Yeh WC. Optimization of the disassembly sequencing problem on the basis of self-adaptive simplified swarm optimization. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*. 2012; 42(1): 250–261.
21. Yeh WC. Novel swarm optimization for mining classification rules on thyroid gland data. *Information Sciences*. 2012; 197: 65–76.
22. Yeh WC. Orthogonal simplified swarm optimization for the series–parallel redundancy allocation problem with a mix of components. *Knowledge-Based Systems*. 2014; 64: 1–12.
23. Bae C, Yeh WC, Wahid N, Chung YY, Liu Y. A new simplified swarm optimization (SSO) using exchange local search scheme. *International Journal of Innovative Computing, Information and Control*. 2012; 8(6): 4391–4406.
24. Clerc M, Kennedy J. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*. 2002; 6(1): 58–73.
25. Ben Ghalia M. Particle swarm optimization with an improved exploration-exploitation balance. *MWSCAS 2008: Proceedings of the 51st Midwest Symposium on Circuits and Systems*; 2008 Aug 10–13; Knoxville, Tennessee. New York: IEEE; 2008. p. 759–762.
26. Kudo M, Sklansky J. Comparison of algorithms that select features for pattern classifiers. *Pattern Recognition*. 2000; 33(1): 25–41.
27. UCI Machine Learning Repository [Internet]. Irvine (CA): the UC Irvine Machine Learning Repository. Available from: <http://archive.ics.uci.edu/ml>
28. Rashedi E, Nezamabadi-Pour H, Saryazdi S. GSA: a gravitational search algorithm. *Information sciences*. 2009; 179(13): 2232–2248.
29. Derrac J, García S, Molina D, Herrera F. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*. 2011; 1(1): 3–18.

30. Fahad A, Alshatri N, Tari Z, Alamri A, Khalil I, Zomaya A, et al. A survey of clustering algorithms for big data: Taxonomy and empirical analysis. *IEEE Transactions on Emerging Topics in Computing*. 2014; 2(3): 267–279.
31. Mitra P, Murthy CA, Pal SK. Unsupervised feature selection using feature similarity. *IEEE transactions on pattern analysis and machine intelligence*. 2002; 24(3), 301–312.
32. Dy JG, Brodley CE. Feature selection for unsupervised learning. *The Journal of Machine Learning Research*. 2004; 5: 845–889.
33. Ng RT, Han J. Clarans: A method for clustering objects for spatial data mining. *IEEE Transactions on Knowledge and Data Engineering*. 2002; 14(5): 1003–1016.
34. Johnson WB, Lindenstrauss J. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary mathematics*. 1984; 26: 189–206.
35. Boutsidis C, Zouzias A, Drineas P. Random projections for *K*-means clustering. In *Advances in Neural Information Processing Systems*. 2010; 298–306.
36. Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters. *Communications of the ACM*. 2008; 51(1): 107–113.
37. Kang U, Tsourakakis CE, Faloutsos C. Pegasus: mining peta-scale graphs. *Knowledge and information systems*. 2011; 27(2): 303–325.
38. Banharsakun A, Sirinaovakul B, Achalakul T. The best-so-far ABC with multiple patrilines for clustering problems. *Neurocomputing*. 2013; 116: 355–366.