Applied Soft Computing 41 (2016) 290–304

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Combining K-Means and K-Harmonic with Fish School Search Algorithm for data clustering task on graphics processing units

Adriane B.S. Serapião*, Guilherme S. Corrêa, Felipe B. Gonçalves, Veronica O. Carvalho

Univ Estadual Paulista (UNESP/IGCE/DEMAC), Rio Claro, São Paulo, Brazil

ARTICLE INFO

Article history:
Received 31 May 2015
Received in revised form
18 November 2015
Accepted 20 December 2015
Available online 6 January 2016

Keywords:
Data clustering
K-Means
K-Harmonic Means
Particle Swarm Optimization
Fish School Search
Graphics processing units

ABSTRACT

Data clustering is related to the split of a set of objects into smaller groups with common features. Several optimization techniques have been proposed to increase the performance of clustering algorithms. Swarm Intelligence (SI) algorithms are concerned with optimization problems and they have been successfully applied to different domains. In this work, a Swarm Clustering Algorithm (SCA) is proposed based on the standard K-Means and on K-Harmonic Means (KHM) clustering algorithms, which are used as fitness functions for a SI algorithm: Fish School Search (FSS). The motivation is to exploit the search capability of SI algorithms and to avoid the major limitation of falling into locally optimal values of the K-Means algorithm. Because of the inherent parallel nature of the SI algorithms, since the fitness function can be evaluated for each individual in an isolated manner, we have developed the parallel implementation on GPU of the SCAs, comparing the performances with their serial implementation. The interest behind proposing SCA is to verify the ability of FSS algorithm to deal with the clustering task and to study the difference of performance of FSS-SCA implemented on CPU and on GPU. Experiments with 13 benchmark datasets have shown similar or slightly better quality of the results compared to standard K-Means algorithm and Particle Swarm Algorithm (PSO) algorithm. There results of using FSS for clustering are promising.

1. Introduction

Over the last few years, new technologies provided a huge increase in storage capacity and data processing, creating a demand for analysis of large data volumes captured by scientific instruments or generated by simulations, resulting in data with a high dimensionality. However, the computational tools for inspection and knowledge extraction from databases not developed in the same level. The cluster analysis (clustering) plays a central role in Data Mining and Knowledge Discovery in Databases, assisting in the data growing problem solving and producing a data separation model to discover groups of similar objects.

In cluster analysis, each group, called cluster, consists of objects that are similar among them and different from the objects of other groups. In general, a dissimilarity measure based on a distance metric is used to define the proximity between a pair of objects. The organization of data in clusters is performed according to a given similitude by using an unsupervised learning approach, with a dataset not labeled, from which it seeks to find out how objects are arranged [1]. In the partitional clustering approach, the algorithms search for determining the clusters' centers (called centroids), according to a fixed number of clusters and a given criterion, in order to produce the best separation among the data. Thus, partitional clustering can be formulated as a global optimization task. The clustering task is computationally difficult (NP-hard), since the partitioning principle of the groups is based on the minimization of intra-group dissimilarity and the maximization of the distance between groups.

K-Means algorithm [2] is the main partitional clustering method, which is very popular because of its computational simplicity. However, it is highly sensitive to the selection of the initial cluster centroids and it is often trapped into local minima due to its hill climbing approach. Similarly to Swarm Intelligence (SI) algorithms, K-Means employs an iterative refinement approach based on the minimization of an objective function. In K-Means, an object belongs only to the closest cluster, the centroids are updated using only their selected cluster objects, and a centroid cannot move out of a local density of data. Thus, the centroid of a cluster can migrate as objects are assigned to it.

* Corresponding author.
E-mail addresses: adriane@rc.unesp.br (A.B.S. Serapião), guil_sanchez_@hotmail.com (G.S. Corrêa), felipe_bonon@yahoo.com.br (F.B. Gonçalves), veronica@rc.unesp.br (V.O. Carvalho).

http://dx.doi.org/10.1016/j.asoc.2015.12.032
1568-4946/

An alternative to solve the drawbacks of the centroids' initialization presented by K-Means is the K-Harmonic Means (KHM) algorithm [3]. But it does not solve the convergence problem to local optima. However, it plays a better role on optimizing performance than K-Means. K-Harmonic Means is based on soft membership (an object belongs to all clusters), dynamic weighting (objects not close to any center are boosted by a higher weight in the next step) and the centroid is updated using all objects weighted by both soft membership and dynamic weighting.

Recently, bioinspired stochastic search methods, especially the algorithms based on population behavior and self-organization of social insects and animals, known as Swarm Intelligence algorithms, have been considered as an emerging alternative to traditional clustering methods, because are very suitable to deal with global optimization problems. These algorithms are effective, robust and adaptive, producing almost optimal solutions through mechanisms of evolution. Numerous studies have been performed concerning clustering using SI [4–30]. Most of them employ Particle Swarm Optimization (PSO) [31–50], the simplest SI algorithm, which is the SI paradigm that has received widespread attention in research.

A usual implementation of SI algorithms for clustering is to evolve a set of possible clusters centroids (candidate solutions) for determining an almost optimal partitioning of a dataset [51]. An important advantage of SI algorithms applied to data clustering task is their ability to handle local minima by recombination and comparison of various candidate solutions simultaneously, enabling the refinement of the final solution.

SI systems are characterized by groups of independent agents that work locally and that collectively interact among them and with the environment to solve a problem. Therefore, they exhibit a large implicit parallelism. In such systems, there is no centralized control structure that decides how agents should behave. Local interactions among agents lead to the emergence of a global behavior. Thus, implementations using parallel processing appear as a good natural way to represent the computational model of swarms. One of the most explored techniques for developing parallel SI algorithms is GPU programming (Graphical Processing Unit) using CUDA (Compute Unified Device Architecture) [52].

An advantage of the implementation of SI algorithms on GPU is that it can increase the population size of individuals and the scale of the problem, greatly accelerating their implementations, providing a feasible solution for complex optimization problems in a reasonable time. Several studies [53–62] have been conducted to parallelization on GPU of SI algorithms for global optimization problems.

Nevertheless, researches involving data clustering analysis, SI algorithms and GPU programming are still scarce in the literature, finding just the work of Situ et al. [63] in this direction. The authors proposed a scheme for the specific task of gene expression clustering analysis, merging K-Means with the particle-pair optimizer (PPO), a variation of the PSO algorithm, and accelerating the performance with GPU. The results were successful in finding better clustering and reducing time consuming.

Based on what stated above, our work explores a framework, Swarm Clustering Algorithm (SCA), for data clustering using SI algorithms, and due to the inherently parallel nature of these algorithms, they were also implemented on GPU for comparison to their serial implementations. In relation to the SI algorithms, it is known that while PSO clustering approaches have been exhaustively explored in the literature, others SI algorithms are not frequently employed for the same purpose yet. Therefore, we target to investigate whether Fish School Search (FSS) algorithm [65–68] is an efficient alternative SI algorithm for multidimensional Data Clustering based on SCA framework and whether it overcomes the drawbacks of PSO for this task. No related work was found in the literature applying FSS to Data Clustering. Furthermore, we also intended to compare the efficiency of FSS, in both implementations, serial and parallel, when used in SCA, with the traditional partitional K-Means clustering method.

Considering the presented context, this paper aims to integrate K-Means and K-Harmonic with two SI algorithms, FSS against the popular PSO, which is used as baseline, in order to investigate the usage of these population based algorithms to the clustering problem with the purpose of improving the efficiency and enhancing the convergence. SI algorithms dispose powerful mechanisms to exploit the search space, raising the chance of reaching optimal clusters partitions. So, the proposed SCA is compared to standard K-Means for evaluation. We have tested 13 well-known benchmark datasets for assessing their clustering quality, using two internal cluster validity measures (SSW and SSB). Besides, comparisons using accuracy and execution time were also considered. The results have shown that the hybrid approach using SI algorithms overcame the standard K-Means algorithm in the clustering task.

An important feature of SCA is its versatility. Its application is not restrict to FSS and PSO, neither to K-Means nor K-Harmonic. Any SI algorithm can be employed as mechanism for searching clusters and any clustering validity criteria can be used as fitness function for evaluation of the swarm. Finally, it is important to mention that this work was inspired by the work of van der Merwe and Engelbrecht [31] that have developed a hybrid model of K-Means and PSO algorithm [64]. The main difference of our work with their work is that they have used a quantization error as fitness function with K-Means to seed the initial swarm.

This work is organized as follows. The revision of the literature about similar works is presented in Section 2. Section 3 introduces the standard K-Means and K-Harmonic algorithms. Section 4 describes the SI algorithms that are employed in this work: PSO and FSS. Section 5 presents the proposed approach for SCA based on K-Means and K-Harmonic Means. Section 6 presents an overview of GPU computing with CUDA and explains about the parallelization of FSS and PSO in SCA. In Section 7 we illustrate and discuss the results of our experiments with serial and parallel implementations of FSS-SCA against PSO-SCA and standard K-Means. Section 8 concludes this work and presents some alternatives for research in the future.

2. Related works

Clustering is an unsupervised technique that has been applied in many areas, including biology, medicine, social sciences, signal processing, business, marketing and economics. Clustering applications include plant and animal classification, disease classification, image processing, pattern recognition, data compression, machine learning and document retrieval. Since clustering techniques have been applied to a wide range of research problems, advanced methods have been intensively investigated either to enhance known clustering algorithms or to create new approaches.

Soft computing methods have been used to bear down the limitations of the partitional clustering methods, such as K-Means. In the last decade, nature inspired optimization techniques were employed for Data Clustering, such as Simulated Annealing (SA) [6], Genetic Algorithms (GA) [7], Tabu Search (TS) [9], Differential Evolution (DE) [14], Gravitational Search Algorithm (GSA) [16,19], Black Hole (BH) algorithm [25], Intelligent Water Drop (IWD) [27], and mostly, swarm based algorithms, as Particle Swarm Optimization [31–50], Ant Colony Optimization (ACO) [4,8], HBMO (Honey Bee Mating Optimization) [5], ABC (Artificial Bee Colony) [10,24], Glowworm Swarm Optimization (GSO) [12], Firefly Algorithm (FA) [15,17,18], Bat Algorithm (BA) [20], Cat Swarm Optimization (CSO)

[21], Wolf Search Algorithm (WSA) [23] and Cuckoo Search (CS) algorithm [11,26,28,30].

The most common measures of variation within a cluster that SI algorithms resort to optimize their fitness functions in are: sum of squared error (SSE), mean of squared error (MSE), quantization error, KHM, Davis–Bouldin Index (DBI) and similarity coefficient. The measures evaluate the quality of a clustering. For easy and quick reference, Table 1 summarizes all these algorithms with different parameters as fitness functions, performance metrics and datasets used.

Researches involving clustering with the SI algorithms, as the ones mentioned in Table 1, are still recent in the literature. Lately there are many others novel population based algorithms being proposed constantly by the scientific community. Meanwhile, PSO has been extensively explored. Many efforts have been done in PSO based clustering, with many researchers proposing different approaches for optimizing clustering challenges.

The great interest in PSO was motivated by having simple and easy implementation, low computational cost, less parameters to adjust and absence of complicated evolutionary operators, like crossover and mutation in GA, less sensitivity to the objectives and the parameters, high quality stable solution, albeit it has shortcomings too. PSO works well with single objective optimization, but not for multi objective problem. When the search space is high (multidimensional data) its convergence speed becomes very slow. Some versions of PSO depend on the choice of the coordinate system. It suffers with fast and premature convergence in mid optimum points. In order to solve these problems several PSO variants have been developed.

Many approaches integrating PSO and K-Means have been proposed to Data Clustering, including improvements to overcome the limitations of both algorithms. The survey on applicability of different PSO variants to Data Clustering is described in Table 2 using the same components of Table 1 and with theirs contributions for PSO based clustering. More detailed reviews about PSO based clustering are reported in Refs. [69–73].

The studies indicated in Tables 1 and 2 have already shown that the hybrid SI algorithms based on K-Means perform more accurately than a traditional K-Means specification. Most part of them uses SSE as fitness function. We include additionally KHM as fitness function for evaluating our proposed algorithms as well.

Based on what stated above, in the present work, we intend to investigate how a hybrid algorithm that combines FSS with K-Means and KHM behaves, both in its serial and parallel processing versions, compared to PSO. Note that FSS was not evaluated for either serial or parallel clustering yet. It was not found any reference in the literature about FSS clustering.

3. K-Means and K-Harmonic Means Algorithms

K-Means is one of the simplest unsupervised learning algorithms for solving the clustering problem, based on a simple iterative scheme for finding a locally minimal solution [2]. This algorithm tags a given dataset defining a number K of clusters fixed a priori. To each cluster is associated a centroid, which is its center of gravity, placed in any location of the problem space. The idea is to split a dataset samples into K groups (clusters), in which each object has common characteristics with the others objects on the same cluster, maximizing the inter-cluster distances and minimizing the intra-cluster distances.

The algorithm starts with a random set of K initial center points of the cluster C_i ($i = 1,, \ldots, K$), which are the current centroids. The first step is to compute the distance (or dissimilarity measure) of each object to each cluster center and associate it to the closest centroid. After that, the next step is to recalculate the K new centroids, resulting from the previous step. These two steps are iteratively repeated until convergence, which could be the assignments of the centroids no longer change or until a finite number of iterations is met.

Regarding the calculation of the distance between objects and clusters (which the most used measure is the conventional Euclidean distance), the aim of the clustering is to optimize the objective function (f):

$$f = \sum_{i=1}^{K} \sum_{\substack{j=1 \\ j \in G_i}}^{N} ||x_j - C_i||^2 \quad (1)$$

where K is the number of clusters, N is the number of objects, x_j is the coordinate of object j, C_i is the coordinate of the cluster i and G_i is the group of objects belonging to cluster i.

The algorithm moves the cluster centers around in space in order to minimize the within-cluster squared distances. For each cluster a new centroid is recomputed by averaging the location of all objects assigned to it. To compute the centroid, it is used Eq. (2):

$$C_i = \frac{1}{|G_i|} \sum_{\substack{j=1 \\ j \in G_i}}^{N} x_j \quad (2)$$

where $|G_i|$ is the number of objects contained in the cluster i.

K-Means produces a separation of the objects into groups from which the metric to be minimized can be calculated. Algorithm I presents the pseudocode for K-Means clustering algorithm.

Algorithm I – Pseudocode of the K-Means algorithm.

```
Place randomly the K cluster centers
While not stop criterion do:
For each object do:
  Compute distance measure to each cluster
  Assign it to the closest cluster
End for
  Recalculate the cluster centers positions (2)
End while
```

K-Means is the most popular clustering algorithm because is very simple, flexible, straightforward, easily implemented with fast execution, measurable and efficient in large data collection. Although K-means is an extensively useful clustering algorithm, it suffers from several drawbacks. The number of clusters K must be known in advanced. The objective function of K-means is not convex and hence it may contain local optima. So, unfortunately the algorithm is prone to getting stuck in local minima (also in local maxima and saddle point) solutions. The efficiency of K-Means algorithm heavily depends on selection of random initial centroids. It is sensitive to noises and outliers. Data clustering is not suitable to clusters with different forms and density. The algorithm cannot be applied in data collection that calculation average is not describable (limited to numerical data).

Another algorithm studied for data clustering is K-Harmonic Means [3], which uses the harmonic average as objective function instead of the Euclidean distance in K-Means algorithm. This feature has the advantage to take into account not only the distance between the analyzed object (x_i) and its cluster center (C_i) but also the distance between the object and all other existing cluster centers (C_c). It is also insensitive to the initialization of the centers.

Let $C = \{C_c | c = 1, \ldots, K\}$ be K centers and $X = \{x_i | i = 1, \ldots, N\}$ be N given data points, the K-Harmonic Means is defined as:

$$\mathrm{KHM} = \sum_{i=1}^{N} \frac{K}{\left(\sum_{c=1}^{K} 1 / \left|x_i - C_c\right|^2\right)} \quad (3)$$

Table 1
Summary of soft computing methods for data clustering.

Paper referred	Algorithm	Fitness function	Performance metrics	Datasets
Kao and Cheng [4]	ACO + K-Means (ACOC)	SSE	Accuracy	Iris, Wine, Vertebral Column, Breast Tissue, 2 artificial sets
Fathian et al. [5]	HBMO	SSE	Fitness, number of fitness evaluation	Iris, Wine, Breast Cancer
Güngör and Ünler [6]	SA + KHM (SAKHMC)	KHM	SSE, CPU time	Iris, Wine, Breast Cancer
Auffarth [7]	GA with self adaptation of the mutation rate	SSE, Mahalanobis with global and with local variance, SVD entropy	Jaccard Index, convergence time	Iris, Wine, Breast Cancer, Ionosphere
Liu [8]	ESacc and Sacc based on ACO	SSE	Fitness, Jaccard, Rand, Folks and Mallows indexes, CPU time	Iris, Wine, Sccds
Yaghini and Ghazanfari [9]	Tabu Search + K-Means (Tabu-KM)	SSE	Fitness, CPU time	Iris, Thyroid
Zhang et al. [10]	ABC	SSE	Fitness, Runtime	Iris, Wine, Thyroid
Goel et al. [11]	CS	DBI	Accuracy	Iris, Satellite Image datasets
Huang and Zhou [12]	GSO + K-Means (GSOCA + KM)	SSE	Accuracy	Iris, artificial data
Karaboga and Ozturk [13]	ABC	SSE	Error rate	Iris, Wine, Breast Cancer, Breast Cancer Int, Credit, Balance Scale, Glass, Heart, Horse, Thyroid, Ecoli, Dermatology, Diabetes
Kwedlo [14]	DE + K-Means (DE-KM)	SSE	Fitness	Iris, TSP-LIB, Image segmentation data
Senthilnath et al. [15]	KFA	SSE	Error rate	Iris, Wine, Breast Cancer, Breast Cancer Int, Credit, Balance Scale, Glass, Heart, Horse, Thyroid, Ecoli Dermatology, Diabetes
Yin et al. [16]	GSA + KHM (IGSAKHM)	KHM	Fitness, F-Measure, Runtime	Iris, Wine, Glass, CMC, Breast Cancer, 2 artificial sets
Abshouri and Bakhtiary [17]	FA + KHM (FFAKHM)	KHM	Fitness, F-Measure	Iris, Wine, Glass, CMC
Hassanzadeh and Meybodi [18]	FA + K-Means (KFA)	SSE	Fitness, Error rate	Iris, Wine, Breast Cancer, Glass, Sonar
Hatamlou et al. [19]	GSA + K-Means (GSA-KM)	MSE	SSE, number of fitness evaluation	Iris, Wine, Glass, CMC, Breast Cancer
Liu et al. [20]	BA	Similarity coefficient	Accuracy	Iris, Wine
Liu et al. [21]	CSO + K-Means (KSACSOC)	SSE	Fitness, Success rate, run time	Iris, Wine, Breast Cancer, Glass, Vowel, Crude Oil, Data-52, Data-62
Tang et al. [23]	WSA + K-Means (C-wolf)	SSE	Fitness	Iris, Wine, Haberman's, Libras, Synthetic, Mouse
Yan et al. [24]	Hybrid ABC with GA crossover operator (HABC)	SSE	Fitness	Iris, Wine, CMC, Breast Cancer, Glass, Liver Disorders
Hatamlou [25]	BH	SSE	Error rate, intra cluster distance, ranking	Iris, Wine, Glass, CMC, Breast Cancer, Vowel
Senthilnath, et al. [26]	CS	SSE	Error rate, time complexity, execution time	Glass, Vehicle, Image segmentation, Crop Type
Shah-Hosseini [27]	IWD + K-Means (IWD-KM)	SSE	Fitness	Iris, Wine, Breast Cancer, Thyroid, Ecoli, Image segmentation
Fong et al. [29]	C-ACO, C-Firefly, C-Cuckoo, C-Bat	SSE	Fitness, CPU time	Iris, Wine, Haberman's, Libras, Synthetic, Image segmentation
Salda et al. [20]	CS	SSE	Fitness, convergence	Iris, Wine, Breast Cancer, Vowel

Table 2
Summary of PSO based data clustering methods.

Paper referred	Algorithm	Fitness function	Performance metrics	Datasets	Contributions
van der Merwe and Engelbrecht [31]	Simple PSO PSO + K-Means	SSE	Fitness	Iris, Breast Cancer, Wine, Automotives	Better convergence to lower SSE, larger SSB, smaller SSW
Ye and Chen [32]	PSO + K-Means with novel metric	Alternative based on inverse exponential of Euclidean distance	Clusters centers	Iris, 6 artificial sets	More robust clustering
Ghali et al. [33]	PSO + exponential inertia weight (EPSO)	Quantization error	Fitness, convergence	Iris, Glass, Breast Cancer, Yeast, Lenses	Slower convergence and lower quantization error
Premalatha and Natarajan [34]	PSO + K-Means + local search based on roulette wheel selection	Quantization error	Fitness, intra and inter cluster distances (with Euclidean and Chebychev)	Iris, Wine, Glass	More accurate and robust clustering
Kao and Lee [35]	Combinatorial PSO + K-Means (KCPSO)	DBI, SSE	DBI, number of clusters found, computational time	Iris, Breast Cancer, Data32,Data43, Data52, Data62	Uses discrete PSO, finds correctly the number of clusters, fast execution
Yang et al. [36]	PSO + KHM (PSOKHM)	Weighted KHM	F-Measure, runtime	Iris, Wine, Glass, CMC, Breast Cancer, 2 artificial set	Improves the convergence speed of PSO, helps KHM escape from local optima
Rana et al. [37]	Hybrid sequential PSO + K-Means (apply PSO and after K-Means)	SSE	Quantization error, inter e intra cluster distances	Iris, Wine, 2 artificial sets	Algorithms in sequence to overcome their drawbacks. More accurate and robust clustering
Chen [38]	PSO + K-Means with stochastic disturbance to control velocity, multidimensional assinchronism (MSPSO_K)	SSE	Fitness	Web log filet from a commercial website	Improves the global search ability to preserve the diversity population of PSO, works well with large scale datasets
Chuang et al. [39]	PSO + Gauss chaotic map to balance the local and global search abilities	SSE	Error rate, intra cluster distance	Iris, Wine, Breast Cancer, Crude Oil, CMC, Vowel	Fastest convergence, highest stability
Neshat et al. [40]	PSO + K-Means	SSE	Fitness, error rate, convergence	Iris, Wine, Breast Cancer, Glass, Sonar, Diabetes	Efficient and robust clustering
Rana et al. [41]	PSO + boundary restricted adaptive for compact clustering (BRAPSO)	SSE	Fitness, error rate, convergence	Iris, Wine, Glass, CMC, Crude Oil, Vowel, 2 artificial sets	Handles particles outside the boundary search space, fast convergence, higher accuracy
Satapathy et al. [42]	PSO + DE Parallel (IPSODEPEA) Transactional (IPSODETEA)	Function based on intra and inter cluster distances	Fitness, quantization error	Iris, Wine, Breast Cancer, Hayes, Diabetes	Better fitness, lower error quantization
El-Tarabily et al. [43]	Hybrid subtractive clustering PSO	SSE	Quantization error, inter e intra cluster distances	Iris, Wine, 2 artificial sets	Fast convergence, lower quantization error, larger SSB, smaller SSW, require less iteration
Kuo and Zulvia [44]	PSO with automatic determination of number of clusters (ACPSO)	given by Gaussian function for specified number of clusters and ratio between intra and inter cluster distances	Fitness, average and standard deviation of number of clusters, accuracy	Iris, Wine, Glass, Aggregation	Number of clusters found is very close to the original number of clusters.
Satapathy et al. [45]	AUTOPSO + dimensionality reduction to reduce the complexity of datasets	Function based on intra and inter cluster distances	Accuracy	Iris, Wine, Breast Cancer, Hayes, Diabetes	Comparable to the algorithm that uses full dimension space, faster runtime
Singh and Singh [46]	K-Means + PSO (improved K-Means)	SSE	Runtime, error rate, accuracy	Breast Cancer, Ecoli, Thyroid	More accurate and efficient K-Means
Lashkari and Rostami [47]	PSO + extended particle's velocity updating	SSE	Convergence speed, number of fitness evaluations, Purity Index, Index of validity	Iris, Glass, Seeds	More suitable convergence speed, lower complexity, higher clustering quality

Table 2 (*Continued*)

Paper referred	Algorithm	Fitness function	Performance metrics	Datasets	Contributions
Saini and Kaur [48]	PSO + K-Means (sequentially applying PSO and K-Means to the search space)	SSE	Quantization error, execution time, inter and intra cluster distances, accuracy	Iris, Wine, artificial set	Better convergence, lower quantization errors, larger inter cluster and smaller intra-cluster distances, same execution time. higher accuracy
Yang and Liu [49]	Fuzzy C-Means (FCM) + PSO (FCM-PSO) FCM-PSO + multi-round random sampling (FC-MPSO-MRS)	Weighted SSE	Accuracy, convergence	Iris, Wine, Glass	Improves initialization procedure of FCM, faster convergence
Zheng et al. [50]	PSO with entropy based fuzzy clustering (EFCPSO)	SSE	Accuracy, fitness	Iris, Wine	Higher accuracy, better stability

The KHM algorithm solves the problem of initialization of the K-Means, but it also easily sensitive to local optima and it acts with specific number of cluster as well.

4. Swarm Intelligence Algorithms

Swarm Intelligence is a set of stochastic techniques based on the collective behavior of self-organizing, distributed, autonomous, flexible and dynamic systems. These systems consist of a population of simple computational agents that have the ability to understand and to modify their local environment. This capability makes possible the communication among the agents, which capture changes in the environment generated by the behavior of their neighbors. Since there is no centralized control structure that establishes how agents should behave, local interactions among agents generally lead to the emergence of a global behavior, even without an explicit model of the environment.

Swarm Intelligence is a branch of Bioinspired Computing that seeks inspiration in the living beings for solving intricate problems. Such algorithms are governed by a principle of constructing an initial population of individuals, followed by local search and interaction among individuals to improve the function related to the ideal solution.

We describe bellow the two swarm algorithms used in this work. The first one, PSO, is the baseline for performance's comparison. The second algorithm, FSS, is the focus of this work.

4.1. Particle Swarm Optimization Algorithm

The underlying motivation for developing the PSO algorithm was the social behavior of organisms such as swarm of bees, herd of animals, and human social behavior [64]. The concept of 'particles' underpins the framework for deploying PSO models, and each particle possesses 'memory' and tracks its coordinates in the D-dimensional problem space. Its 'velocity' is dynamically adjusted to 'experience' and 'global success'. Experience represents the fact that the PSO model tries out some choices, and then this model is able to identify the best state ('p_B'), and does recognize the previous 'goodness' of this state ('fitness'). Global success ('g_B') is related to the particles that constitute the neighborhood of the particle under consideration. That is, the performance of the other particles is assessed, so that the PSO model is aware of the choices of the neighboring particles.

PSO also involves regulating the velocity ($\overrightarrow{v_i}$) and the position ($\overrightarrow{x_i}$) vectors of each particle i (with $i = \{1, 2, \ldots, n\}$ and n = population size) at the current state t on the D-dimensional space with the purpose of reaching its p_{B_i} and g_B locations as follows:

$$\overrightarrow{v_i}(t+1) = \omega \times \overrightarrow{v_i}(t) + \varphi_1 \times (p_{B_i} - \overrightarrow{x_i}(t)) + \varphi_2 \times (g_B - \overrightarrow{x_i}(t)) \quad (4)$$

$$\overrightarrow{x_i}(t+1) = \overrightarrow{x_i}(t) + \overrightarrow{v}(t+1) \quad (5)$$

in which φ_1 and φ_2 stand for the two random functions that are uniformly distributed within the interval [0,1]. The positive constants φ_1 and φ_2 are termed cognitive rate and social learning rate respectively. ω represents the inertia weight, which is used with the purpose of limiting the velocity, p_{B_i} is the coordinates of best solution found by the particle i up to time t and g_B is the best position ever found by the total population.

Algorithm II introduces the pseudocode for PSO.

Algorithm II – PSO Pseudocode.

```
Set the initial paramenters ω, φ2 and φ2
Generate the initial population of n particles (x_i)
Generate the initial velocity of n particles (v_i)
While not stop criterion do:
  For each particle do:
Calculate the fitness
    Calculate its best position up to now (p_B)
End for
Determine the best particle (g_B)
For each particle do:
Update the velocity (4)
    Update the position (5)
End for:
  Find the best particle
End while
```

4.2. Fish School Search Algorithm

FSS is an optimization algorithm proposed by Bastos Filhos et al. [65,66] that takes inspiration from behavior of schools. FSS relies on the feeding and swimming of fishes in an aquarium (hyper dimensional search space). Each fish features a candidate solution for the problem. Fishes have a weight (W) and a position vector ($\overrightarrow{x}$). Fishes are attracted to food scattered in the aquarium in various concentrations. In order to find greater amounts of food, the fish in the school can move independently. Therefore, each fish is allowed to grow in weight, depending on its success or failure in obtaining food. The feeding operator represents the regions of the aquarium, which are likely good spots for finding food during the search process. This operator updates the fish weight according to the quality of motion (i.e. after swimming). The weight indicates the success of a fish (finding food) and it changes at each iteration (t). In other words, the feeding operator measures the quality of a solution.

In this algorithm, swimming is considered to be an elaborate form of reaction regarding survivability. The swimming operators guide globally the fishes toward the aquarium. There are three types of swimming movements that a fish can execute [67]:

- individual movement, where the fish moves in its neighborhood, which is determined by a swimming step (fish displacement in the aquarium);
- collective instinctive movement, where all fishes move in accordance with an overall direction, which is computed by the weighted average of individual movements based on the instantaneous success of all fishes of the school;
- collective volitive movement, which is framed as an overall success/failure evaluation, according to the incremental weight variation of the whole fish school.

The equations driving the individual movement are described as:

$$\vec{x_i}(t+1) = \vec{x_i}(t) + rand(-1, 1) step_{ind}(t) \quad (6)$$

$$\Delta f_i = f(\vec{x_i}(t+1)) - f(\vec{x_i}(t)) \quad (7)$$

$$\Delta \vec{x_i} = \vec{x_i}(t+1) - \vec{x_i}(t) \quad (8)$$

where: $\vec{x_i}$ is the current position of the fish i ($i = \{1, 2, \ldots, n\}$, with n = school size) at the D-dimensional space; t is the current iteration; $rand(-1, 1)$ is a random number uniformly generated in the interval $[-1, 1]$; $step_{ind}$ is a number that decreases after each iteration; and f is the fitness function, that indicates the amount of food at the location $\vec{x_i}$.

The $step_{ind}$ is a percentage of the search space amplitude and decreases linearly during iterations according to Eq. (9) in order to improve the exploitation ability in later iterations:

$$step_{ind}(t+1) = step_{ind}(t) - \frac{(step_{ind\ \ initial} - step_{ind\ \ final})}{iterations} \quad (9)$$

where $step_{ind\,initial}$ and $step_{ind\,final}$ are predefined and represent, respectively, the initial and the final individual movement step.

All fishes are born with weight equal. The feeding operator is computed as:

$$W_i(t+1) = W_i(t) + \frac{\Delta f_i}{\max(\Delta f)} \quad (10)$$

where W_i is the weight of the fish i; Δf_i is the difference of fitness (objective-function of the problem) calculated as Eq. (7) in the individual movement; and $\max(\Delta f)$ is a function that returns the maximum difference of the fitness values among all the fishes.

Fishes that had successful individual movements (growing weight) influence the resulting direction of the school movement more than the unsuccessful ones. The resulting direction vector ($\vec{I}$), which indicates the direction that all fishes will move, is calculated by:

$$\vec{I}(t) = \frac{\sum_{i=1}^{N} \Delta \vec{x_i}\, \Delta f_i}{\sum_{i=1}^{N} \Delta f_i} \quad (11)$$

where N is the number of fishes, $\Delta \vec{x_i}$ (Eq. (8)) is the displacement given by the previous and current positions.

After the overall direction is computed, each fish is repositioned. This collective instinctive movement is calculated as:

$$\vec{x_i}(t+1) = \vec{x_i}(t) + \vec{I}(t) \quad (12)$$

Based on the overall performance of the fish school at each iteration (t), the collective volitive movement occurs. This movement considers the following: if the fish school is putting on weight (success), the radius of the school should contract; if not (failure), it should dilate. In order to enhance the exploration abilities, when the whole school is achieving better performance, the operator approximates the fish aiming to accelerate the convergence toward a good region. Otherwise, the operator spreads the fish away from the barycenter of the school and the fish has more chances to escape from a local minimum. The fish school expansion or contraction is applied as a small step drift to every fish position regarding the school barycenter ($\vec{B}$), which can be evaluated as:

$$\vec{B}(t) = \frac{\sum_{i=1}^{N} \vec{x_i} w_i(t)}{\sum_{i=1}^{N} w_i(t)} \quad (13)$$

In order to perform the fish school contraction (sign −) or expansion (sign +), the volitive movement is respectively obtained from:

$$\vec{x_i}(t+1) = \vec{x_i}(t) - step_{vol} rand(0, 1) \frac{(\vec{x_i}(t) - \vec{B}(t))}{distance(\vec{x_i}(t), \vec{B}(t))} \quad (14)$$

$$\vec{x_i}(t+1) = \vec{x_i}(t) + step_{vol} rand(0, 1) \frac{(\vec{x_i}(t) - \vec{B}(t))}{distance(\vec{x_i}(t), \vec{B}(t))} \quad (15)$$

where: $distance(\,)$ is a function which returns the Euclidean distance between the barycenter and the fish current position; $step_{vol}$ is a predetermined step used to control the displacement from/to the barycenter; and $rand(0,1)$ is a number randomly generated by an uniform distribution in interval [0,1].

All fishes must update their positions according to Eq. (14) if the weight of the school (sum of all fishes) increased at the current iteration or according to Eq. (15) if the weight of the school decreased at the current iteration.

Algorithm III illustrates the FSS pseudocode, in which every movement is governed by the mathematical equations described from Eqs. (6) to (15).

Algorithm III – FSS Pseudocode.

```
Set the initial paramenters step_initial, step_final and step_vol
Generate randomly the initial population of n fishes (x_i)
  and weight (W_i)
While not stop criterion do:
For each fish do:
Calculate the fitness
Execute the individual movement with (6)–(8)
Feed the fish with (10)
End for
Calculate the direction vector with (11)
For each fish do:
Execute the instinctive movement with (12)
End for:
Calculate barycenter with (13)
For each fish do:
Execute the volitive movement with (14) and (15)
End for
Update step_ind with (9)
End while
Find the best fish
```

5. Swarm Clustering Algorithm

This work proposes to combine SI algorithms with traditional partitional clustering methods, K-Means and KHM, forming a framework that was named Swarm Clustering Algorithm, taking advantages of the best characteristics of both population based algorithms and classical clustering methods.

Basically, as discussed earlier, the K-Means algorithm consists of two main steps: (i) the calculation of intra-cluster distances and (ii) the update of clusters centers. KHM is based on intra and inter-cluster similarities. Therefore, the way to join the K-Means (or KHM) and the SI algorithms is to make associations between them. The intra-cluster measure can be thought of as how effective these clusters are coordinated to the problem of data clustering. Thus, a dissimilarity measure (K-Means or KHM) is a potential fitness function to be minimized by the swarm algorithms. In this manner, each

$$\overbrace{1,2,3,\ldots D}^{Cluster\ 1}\ \overbrace{1,2,3,\ldots D}^{Cluster\ 2}\ \overbrace{1,2,3,\ldots D}^{Cluster\ 3}\ \ldots\ \overbrace{1,2,3,\ldots D}^{Cluster\ K}$$

Fig. 1. Individual's structure for swarm clustering.

individual of the swarm algorithm will be a possible solution of the clustering problem. Each individual contains the coordinates of all potential centroids of the K clusters.

Thus, the goal of this approach is concerned to find the centroids of K clusters, by exploring the solution space, moving the individuals through the space, which are governed by the movement equations of the SI algorithms. Therefore, each individual stores an array ($K \times D$), where K is the number of clusters and D is the number of dimensions of the cluster (i.e., number of attributes of the dataset), as shown in Fig. 1. The array contains the coordinates values in the D-dimensional Euclidean space for each cluster.

To summarize, the dataset is first loaded in a convenient structure. Initially, each individual of the population (swarm) is put in random locations (representing potential centroids for K-Means, where clusters are randomly distributed). Then, associate each data object to a cluster of each individual. Following, the fitness of these individuals is calculated, which represents the contribution of the current dissimilarity measure (i.e., Euclidean distance or KHM). Next, the movements of the individuals are executed, ruled by the equations of the swarm algorithms, so that the centroids are updated. This procedure is executed until the stop criterion is reached. The best individual (with the best fitness value) contains the best coordinates for the centroids of all clusters. Upon execution, it is still made a post-processing to better visualize the results. The proposed approach, named Swarm Clustering Algorithm, is closely related to the proposed one by van der Merwe and Engelbrecht [31]. The authors proposed a PSO Clustering Algorithm using K-Means clustering to seed the initial swarm and the quantization error as fitness function, with the particles representing the cluster centroid vectors. We have expanded this idea for using any SI algorithm and any fitness function.

The general SCA is described on the Algorithm IV.

Algorithm IV – Pseudocode of the General Swarm Clustering Algorithm.

```
Dataset loading
Set all initial parameters
Generate randomly the initial population with n individual
  composed by K centroids
While not stop criterion do:
For each individual do:
Assign it to the closest cluster
  End for
  For each individual do:
    Calculate fitness
  End for
For each individual do:
    Apply swarm movements
    Update centroids
  End for
End while
Best Individual has the optimal centroids
```

For illustrating the behavior of SCA, Fig. 2 shows a hypothetical dataset, represented in a bidimensional space ($d = 2$) (attributes x and y), as black circles. The dataset must be split in three clusters ($K = 3$). Black squares, triangles, diamonds, times, pentagons and stars symbols represent individuals of the swarm ($n = 6$). At the first step, the population is randomly disposed in the problem space (Fig. 2a). Each individual (I_i, with $i = \{1, 2, \ldots, n\}$) is composed by a structure as shown in Fig. 1, which contains the x-y values of each initial centroids. So, the swarm is represented by a $n \times (d \times K)$ matrix (Fig. 2d). The role of swarm is to scan the space problem to detect the best centroids set. The greater the number of individuals, the greater the exploration of the problem space will be. During the

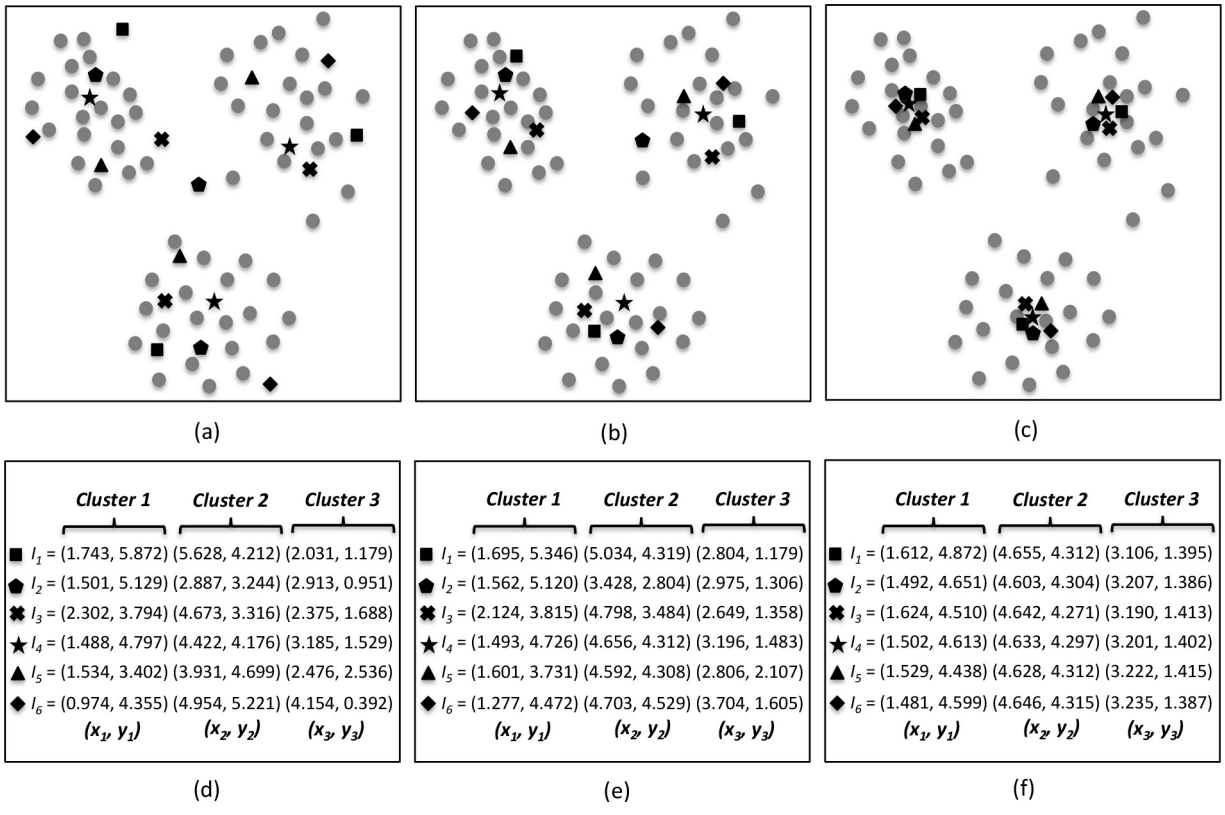

Fig. 2. Swarm clustering movements. Gray circles are objects of the dataset and the other black symbols are potential solutions (individuals of population). (a) Random initial population. (b) Evolving population some iterations after. (c) Evolved population at the last iteration (final solution). (d), (e) and (f) are potential centroids of the population in (a), (b) and (c), respectively.

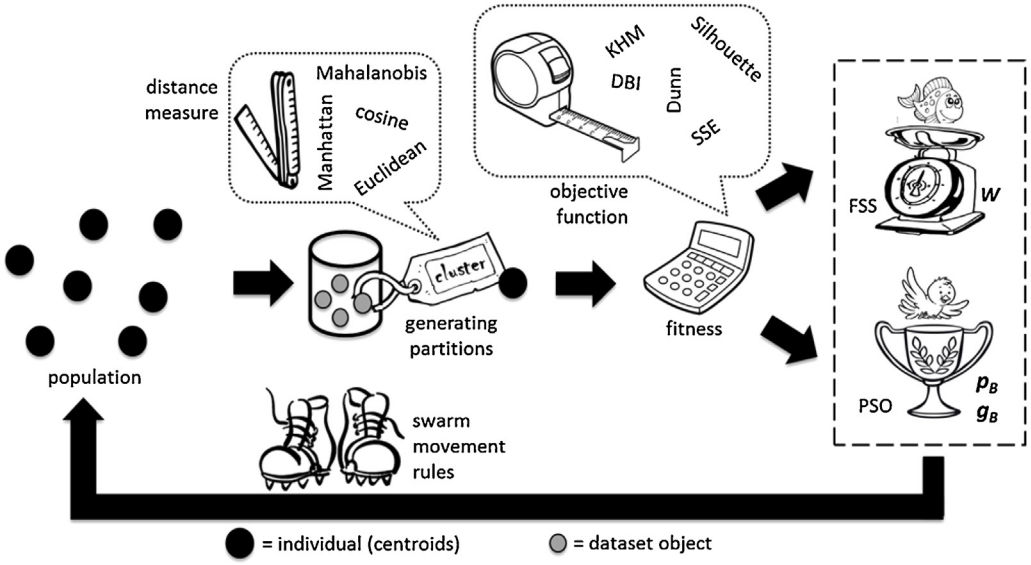

Fig. 3. Aggregating clustering measurements to SI algorithms.

swarm evolving (Fig. 2b), each individual incorporates both information about its own navigation and the population, and adjusts its location (Fig. 2e), traveling through the space, governed by the swarm movement equations, based on the best evaluations of the centroids (given by fitness function). At the end of the evolving process, the most part of individual should migrate for K regions of the space (Fig. 2c), concentrating the population in some locations, indicating the promising best centroids (Fig. 2f). The best solution is given by the individual that contains the highest fitness.

Fig. 3 exhibits the way that the clustering dissimilarity measures should be used and how they guidance the swarm movements in the SCA. The population (centroids) is randomly created. First, each object from dataset is assigned to the closest cluster, according to a distance measure (Euclidean, Mahalanobis, Manhattan, Chebychev, cosine, etc.), generating the partitions. The performance of the individuals reaching the problem solution is evaluated by the fitness function, which for PSO is associated to *global success* (g_B) and *local success* (p_B), and to *weigh* (W) for FSS. For others SI algorithms, the equivalent performance criterion of individuals must be taken into account. Sometimes, the fitness function is equal to the objective function, which defines a measurement for evaluating the solution, as in the case of PSO. For others SI algorithms, the objective function can be used to compose the element that estimates the individual success, as in FSS (Eq. (10)). For the clustering task, a dissimilarity measure (i.e., SSE and KHM) is the objective function that is used in SI algorithms. In this sense, other clustering measurements, like Silhouette Coefficient, DBI and Dunn Index, could be employed as objective function to integrate the fitness function of SCA as well.

The fitness function is also employed to drive the movement of the individuals, allowing their wandering through space and updating their locations (using Eqs. (4)–(5) for PSO and Eqs. (11)–(15) for FSS). So, the centroids (potential solutions) are recalculated. This procedure is repeated until a stop criterion is met.

6. GPU Swarm Intelligence Algorithms Implementation

GPUs are notably well fitted to deal with problems that can be depicted as data parallel computations, where the same repertoire of instructions can be carried out in parallel on various data processing units. Data parallel processing forges such data units to parallel processing threads. In this section, we introduce the principles of GPU computing and give details of the parallel SI algorithms programming.

6.1. General Purpose GPU Computing

The intensive usage of compute-capable graphics processing units (GPUs) [52] in personal computers and workstations becomes them very interesting as accelerators for high-performance parallel computing. Such processing power, combined with the large capacity at high speeds data transmission, has lead researchers and scientists to examine the use of these video cards not only for their original purpose, but also for general computing. Thus, this technology produced video cards known as General Purpose Graphic Processing Unit (GPGPU).

In order to use the power of GPGPU in computational problems, CUDA (Compute Unified Device Architecture) was developed by NVIDIA as a framework for GPU programming that supports C-like language constructs. The basic idea is to divide scheduling threads in a process, which are executed uniquely in each GPU processor. Unlike a CPU, which has at most a few tens of processor elements, a GPU may have hundreds of these.

A thread can be defined as the smallest control unit to be processed into a core. A group of threads form a block and several blocks form a grid. This scheme of organization is very useful for the programmer to control operations and data flow. For example, within each block there is a local memory shared by all threads of this block. The same occurs with blocks of a grid. In this work, CUDA programs were implemented through "C command extensions". For example, if before the C function declaration is inserted the "__global__" keyword, it indicates that the function is a kernel–that is, a function that will be called from CPU and executed by the GPU. The "__device__" keyword denotes a function that will be executed by the GPU but can only be called from the GPU. The "__host__" keyword is optional. It designates a function that is called from CPU and executed by the CPU.

To work with CUDA it is necessary to have great command of the concept of pointers, because before each kernel call, you must first copy the data of the CPU memory of the process for the GPU, and after the end of the calculations, the result should be copied back. This involves manipulation of pointers, as well as the use of the specific function "cudaMemCpy".

Each thread has a unique number that identifies it, and this number can be used as a counter to multithreaded processes. The call of a kernel, after including the function name, you must also include the parameter '⋘N, M⋙" indicating that the kernel will be divided into "*N*" blocks, and each block with "*M*" threads.

6.2. Implementation details

For the parallel implementation on GPU of FSS and PSO in SCA using CUDA, it was necessary to rearrange both algorithms, so that the data stream were the most efficient, without significant loss of accuracy, due to structural and logical changes to the original algorithms.

The SI algorithms use random numbers intensively in their procedures, which can be very time-consuming. The performance of the optimization leans on the quality of randomization. But for this randomization could be made in parallel, a device for generating a number of seeds equal to the random number of parallel threads in the kernel is required. Therefore, once a run is necessary to call the "setup_kernel" function, which will keep a vector (one-dimensional or two-dimensional) with all generated seeds. These must be updated every time that is used to generate a random number within a parallel function on a GPU CUDA core. Hence, it is possible to obtain *N* random numbers and to use them in *N* parallel threads. In this work, the CURAND library [74] was used for generating random numbers on the GPU.

To choose which processes could be parallelized, it was analyzed those without parameter dependence among them, and that were repeated several times in loop. Therefore, all possible function calls, that were made within a repeating loop, were conveniently changed, in order to be kernels of the new parallelized algorithm.

In a parallel call of CUDA, first it must be copied from Host (CPU) to Device (GPU) any data structure that will be handled in parallel process. If there is only a one-dimensional vector of size *N* containing the information of all the individuals, its size is defined as variable "size_t". It allocates memory from CPU and GPU ("cudaMalloc"), a pointer points to the data structure, and it will be copied to the dynamically allocated memory on GPU through "cudaMemcpy".

Finally, the function calls are performed. An additional C syntax informs how many blocks and how many threads the kernel will be divided. Finally, the data structure returns from CPU to GPU with the reverse "cudaMemcpy" and the allocated memory on the GPU heap is disposed with "cudaFree".

We can access information from blocks and threads of each parallel call via the controls "blockIdx.x" and "threadIdx.x", respectively. This is very useful to control the data flow. It is of utmost importance for high performance in a parallelization with CUDA to maintain a minimum of data flow between Host and Device. That is, it must use the least possible "cudaMemcpy", because it is time consuming, and its time increases exponentially depending on the size of the data structure to be copied and returned from the GPU.

Furthermore, where possible, it is quite advantageous to call kernels involving blocks and threads, instead of just threads, because the parallelization is more comprehensive and effective, since there is a good flow control over undesirable effects that may occur, such as overlapping data or threads that are waiting for the results of other unfinished threads.

In the parallel implementation, in order to reduce communication time among individual population members (and consequently among GPU cores), it was introduced asynchronous operations into the SI algorithms. In asynchronous processing, the GPU cores run entirely independently for a specified number of iterations, after which they are resynchronized.

7. Results and discussions

The results obtained with FSS for clustering were compared with the standard K-Means and PSO clustering. For both, the assignment of objects to the closest cluster we have used Euclidean distance. We have analyzed two measures as fitness function for the SCAs: SSE (as in K-Means algorithm) and K-Harmonic. For sake of simplicity, we denoted the algorithms combined with the measures by: K-FSS and K-PSO when based on K-Means, and KH-FSS and KH-PSO when using K-Harmonic Means. We have also contrasted the results of the serial implementations with their parallel implementations in GPU. The evaluation of computing execution time was considered as well, and the speed-up calculated.

7.1. Experiments

The swarm algorithms were coded in C Language and run on CPU Intel i7 3.20 GHz with 8 GB RAM capacity and NVidia GTX 460 graphics card with 336 cores using CUDA and internal memory of 1GB. Standard K-Means was obtained from Data Mining software WEKA with default parameters. Euclidian distance was used as fitness function and 500 iterations.

For executing each SCA, the parameters' settings were defined as following: (a) PSO has fixed $\omega = 1$ and $\varphi_1 = \varphi_2 = 2.05$; (b) FSS has adopted $step_{initial} = 0.1$, $step_{final} = 10^{-7}$ and $step_{vol} = 0.1$. All experiments considered a population of 75 individuals and 2000 iterations as stop criterion. These parameters were chosen after experimentation varying the values to get the best clustering.

For testing the SCAs, 13 labeled benchmark UCI datasets [75] have been used to perform comparative analysis, including different number of clusters (*K*), number of attributes (dimensions) (*d*) and samples (*n*), in order to obtain better information about the performance of the algorithms. These datasets are commonly used in

Table 3
Evaluated datasets.

Dataset	Number of clusters	Number of attributes	Number of objects	Number of objects by cluster	Type of attributes
Iris	3	4	150	50-50-50	(4 R)
Breast Cancer	2	9	683	444-239	(9 I)
Glass	6	9	214	70-78-17-13-9-29	(9 R)
Wine	3	13	179	59-71-49	(11 R), (2 I)
Olive	9	8	572	25-56-206-36-65-33-50-50-51	(8 I)
Crude oil	3	5	56	7-11-38	(5 R)
Diabetes	3	6	144	33-36-75	(1 R), (5 I)
Egyptian Skulls	3	4	90	30-30-30	(4 R)
Heart Statlog	2	13	270	150-120	(6 R), (1 I), (3 B), (3N)
Ionosphere	2	34	351	126-225	(34 R)
Vehicle	4	18	846	240-240-240-226	(18 R)
Balance Scale	3	4	625	49-288-288	(4 I)
Sonar	2	60	20	111-97	(60 R)

Table 4
Results of the Serial Swarm Clustering Algorithms compared to Standard K-Means.

Dataset	Algorithm	Average			Best run		
		Accuracy (%)	SD	*P*	Accuracy (%)	SSW	SSB
Iris	K-FSS	90.00	**0.0000**	0.0000	90.00	**0.6444**	6.5828
	KH-FSS	**91.87**	0.0062	0.0041	**92.67**	0.6542	6.1751
	K-PSO	81.07	0.0947	0.0627	83.07	0.8511	6.6386
	KH-PSO	91.07	0.0138	0.0091	91.33	0.6556	6.1570
	K-Means	88.67	0.0000	0.0000	88.67	0.6481	**6.7439**
Breast	K-FSS	96.14	**0.0000**	0.0000	96.14	**4.3407**	28.1624
Cancer	KH-FSS	**96.29**	0.0004	0.0003	**96.42**	4.3532	26.5487
	K-PSO	95.08	0.0092	0.0061	96.28	4.7722	**32.6129**
	KH-PSO	82.78	0.2244	0.1485	95.14	4.6571	31.4159
	K-Means	95.71	**0.0000**	0.0000	95.71	4.3816	27.5075
Glass	K-FSS	**50.79**	**0.0062**	0.0041	**51.87**	1.2125	14.0155
	KH-FSS	40.84	0.0358	0.0237	45.33	1.3025	4.4150
	K-PSO	46.54	0.0303	0.0201	47.54	1.6133	**18.3564**
	KH-PSO	45.14	0.0422	0.0279	47.20	1.6759	8.5459
	K-Means	43.73	0.0000	0.0000	43.73	**1.0789**	7.7139
Wine	K-FSS	**71.91**	**0.0000**	0.0000	**71.91**	**91.5311**	897.7753
	KH-FSS	**71.91**	**0.0000**	0.0000	**71.91**	92.6537	831.0958
	K-PSO	70.90	0.0036	0.0024	70.79	93.0117	**914.6921**
	KH-PSO	**71.91**	**0.0000**	0.0000	**71.91**	94.5082	831.3142
	K-Means	71.35	0.0000	0.0000	71.35	99.8227	788.0840
Olive	K-FSS	58.23	0.0552	0.0365	**69.18**	153.4882	**1614.0168**
	KH-FSS	50.96	0.0482	0.0319	58.32	160.9287	1086.4112
	K-PSO	59.21	0.0502	0.0332	60.42	151.4534	1249.7340
	KH-PSO	51.58	**0.0462**	0.0306	53.94	166.7637	1124.1130
	K-Means	**60.42**	0.0000	0.0000	60.42	**130.5072**	1225.5900
Crude Oil	K-FSS	**70.91**	**0.0000**	0.0000	**70.91**	**4.9171**	40.2126
	KH-FSS	58.18	**0.0000**	0.0000	58.18	5.3552	36.0781
	K-PSO	70.18	0.0094	0.0062	**70.91**	4.9618	**40.4395**
	KH-PSO	58.00	0.0415	0.0275	61.00	5.6042	30.6129
	K-Means	65.45	0.0000	0.0000	65.45	5.5187	32.0909
Diabetes	K-FSS	86.71	**0.0033**	0.0022	**87.41**	**127.7354**	1125.9033
	KH-FSS	**86.92**	0.0066	0.0044	**87.41**	138.0545	915.3193
	K-PSO	75.52	0.1179	0.0780	78.52	135.2304	**1444.2330**
	KH-PSO	86.85	0.0098	0.0065	**87.41**	138.0493	913.4683
	K-Means	70.63	0.0000	0.0000	70.63	173.6288	662.4440
Egyptian	K-FSS	40.45	**0.0000**	0.0000	40.45	**6.2722**	**18.5157**
Skulls	KH-FSS	**45.95**	0.0099	0.0066	**47.19**	7.2732	6.2974
	K-PSO	40.79	0.0054	0.0036	40.88	**6.2723**	18.5008
	KH-PSO	43.60	0.0197	0.0130	43.82	7.3251	5.5974
	K-Means	43.82	0.0000	0.0000	43.82	6.8913	18.1717
Heart	K-FSS	60.97	**0.0000**	0.0000	60.97	**39.3063**	141.7939
Statlog	KH-FSS	60.59	0.0018	0.0012	60.97	41.0107	86.8782
	K-PSO	60.74	0.0047	0.0031	60.81	39.6568	**143.2714**
	KH-PSO	60.45	0.0019	0.0013	60.59	41.1560	87.1915
	K-Means	**61.71**	0.0000	0.0000	**61.71**	45.8743	38.5485
Ionosphere	K-FSS	57.77	0.0468	0.0310	66.00	2.5863	6.6191
	KH-FSS	67.92	**0.0309**	0.0205	**72.00**	2.8259	3.8299
	K-PSO	63.40	0.0417	0.0276	66.29	2.9295	**8.0422**
	KH-PSO	63.94	0.0629	0.0416	67.43	2.9878	3.5173
	K-Means	**71.14**	0.0000	0.0000	71.14	**2.2741**	6.1489
Vehicle	K-FSS	45.03	0.0049	0.0032	45.68	**54.1701**	**571.1310**
	KH-FSS	**45.76**	**0.0042**	0.0028	**46.63**	58.4237	469.2079
	K-PSO	40.73	0.0242	0.0160	41.33	71.1079	475.4760
	KH-PSO	38.60	0.0273	0.0181	39.60	93.5622	481.8672
	K-Means	36.80	0.0000	0.0000	36.80	62.8879	464.7720
Balance	K-FSS	51.28	**0.0000**	0.0000	51.28	**2.2760**	**5.4570**
Scale	KH-FSS	52.66	0.1031	0.0682	**76.92**	2.7439	0.0620
	K-PSO	52.71	0.0180	0.0119	53.85	2.2771	5.4158
	KH-PSO	51.79	0.0817	0.0541	52.79	2.7475	0.0385
	K-Means	**53.69**	0.0000	0.0000	53.69	2.2768	5.4499
Sonar	K-FSS	53.14	**0.0039**	0.0026	53.62	1.4911	3.6290
	KH-FSS	53.77	0.0446	0.0295	**64.25**	1.7768	3.9419
	K-PSO	52.22	0.0134	0.0089	53.14	1.4037	**4.2519**
	KH-PSO	**56.47**	0.0580	0.0384	58.47	1.4495	2.2439
	K-Means	54.59	0.0000	0.0000	54.59	**1.1309**	2.4977

the literature for data clustering evaluating and they allow checking the correct assignments of objects to clusters after SCA training. The selected datasets are described in Table 3, that includes the number of objects for each cluster and the type of attributes, classified as real (R), integer (I), binary (B) and nominal (N). For SCA processing, the nominal attributes were converted to integer values.

7.2. Results

In Table 4, as experimental results, we present the average and best accuracy (as external quality criterion) for clustering, as well as standard deviation (*SD*) and 95% confidence interval (*P*) upon 10 independent runs of each algorithm for each dataset using the

Table 5
Results of the Parallel Swarm Clustering Algorithms compared to standard K-Means.

Dataset	Algorithm	Average			Best run		
		Accuracy (%)	SD	*P*	Accuracy (%)	SSW	SSB
Iris	K-FSS	89.87	0.0028	0.0019	90.00	0.6462	6.6744
	KH-FSS	**91.60**	0.0123	0.0081	**93.33**	0.6549	6.1920
	K-PSO	90.00	**0.0000**	0.0000	90.00	0.6444	6.5841
	KH-PSO	91.33	**0.0000**	0.0000	91.33	**0.6558**	6.1549
	K-Means	88.67	0.0000	0.0000	88.67	0.6481	**6.7439**
Breast Cancer	K-FSS	96.14	**0.0000**	0.0000	96.14	**4.3423**	28.4326
	KH-FSS	96.28	**0.0000**	0.0000	96.28	4.3527	26.8566
	K-PSO	96.02	0.0041	0.0027	96.28	4.4949	**29.0437**
	KH-PSO	**96.35**	0.0012	0.0008	**96.42**	4.3681	27.0059
	K-Means	95.71	0.0000	0.0000	95.71	4.3816	27.5075
Glass	K-FSS	50.51	**0.0060**	0.0040	51.40	1.2481	**13.4663**
	KH-FSS	42.80	0.0566	0.0375	**52.80**	1.4760	7.7676
	K-PSO	**51.31**	0.0076	0.0050	52.34	1.1739	13.3666
	KH-PSO	41.64	0.0387	0.0256	47.66	1.2993	4.8833
	K-Means	43.73	0.0000	0.0000	43.73	**1.0789**	7.7139
Wine	K-FSS	71.46	0.0083	0.0055	**72.47**	96.9441	835.3869
	KH-FSS	71.24	0.0121	0.0080	**72.47**	95.9439	795.5167
	K-PSO	71.74	0.0027	0.0018	71.91	**91.5856**	**898.8526**
	KH-PSO	**71.91**	**0.0000**	0.0000	71.91	92.6579	830.1719
	K-Means	71.35	0.0000	0.0000	71.35	99.8227	788.0840
Olive	K-FSS	49.26	0.0637	0.0422	56.92	253.5142	**1681.4959**
	KH-FSS	55.38	0.1039	0.0688	**70.58**	246.8066	1635.4222
	K-PSO	**64.40**	0.0438	0.0290	68.48	**117.4233**	1598.6591
	KH-PSO	51.31	**0.0343**	0.0227	55.34	134.7555	1170.7481
	K-Means	60.42	0.0000	0.0000	60.42	130.5072	1225.5900
Crude Oil	K-FSS	**70.91**	**0.0000**	0.0000	**70.91**	4.9175	**40.2463**
	KH-FSS	58.18	**0.0000**	0.0000	58.18	5.3652	36.1887
	K-PSO	69.45	0.0077	0.0051	**70.91**	**4.9171**	40.1921
	KH-PSO	60.00	0.0514	0.0340	65.45	5.1494	36.0332
	K-Means	65.45	0.0000	0.0000	65.45	5.5187	32.0909
Diabetes	K-FSS	71.05	0.0510	0.0338	76.22	195.2187	1111.4522
	KH-FSS	74.41	0.0521	0.0345	83.92	193.1975	695.8933
	K-PSO	82.24	0.0944	0.0625	86.71	**127.6026**	**1133.0903**
	KH-PSO	**87.13**	**0.0059**	0.0039	**87.41**	137.9827	913.6780
	K-Means	70.63	0.0000	0.0000	70.63	173.6288	662.4440
Egyptian Skulls	K-FSS	40.45	**0.0000**	0.0000	40.45	6.2723	**18.5085**
	KH-FSS	42.47	0.0128	0.0085	44.94	7.2586	6.2445
	K-PSO	40.56	0.0036	0.0024	41.57	6.2750	18.3739
	KH-PSO	42.02	0.0142	0.0094	**46.07**	**7.2631**	6.3455
	K-Means	**43.82**	0.0000	0.0000	43.82	6.8913	18.1717
Heart Statlog	K-FSS	56.54	0.0626	0.0414	60.59	39.3560	137.0998
	KH-FSS	59.03	0.0283	0.0187	**63.94**	47.8002	102.1344
	K-PSO	60.97	**0.0000**	0.0000	60.97	**39.3061**	**141.8378**
	KH-PSO	60.59	**0.0000**	0.0000	60.59	41.0208	86.7255
	K-Means	**61.71**	0.0000	0.0000	61.71	45.8743	38.5485
Ionosphere	K-FSS	70.40	0.0072	0.0048	**71.43**	2.3524	**6.5882**
	KH-FSS	68.14	0.0096	0.0064	69.14	2.3753	4.3701
	K-PSO	69.77	0.0046	0.0030	70.29	2.3699	5.9956
	KH-PSO	67.26	**0.0036**	0.0024	67.43	2.3164	4.4159
	K-Means	**71.14**	0.0000	0.0000	71.14	**2.2741**	6.1489
Vehicle	K-FSS	40.35	0.0110	0.0073	42.60	96.1788	**772.0775**
	KH-FSS	40.01	0.0228	0.0151	45.68	119.6204	512.0215
	K-PSO	43.28	0.0134	0.0089	44.97	**53.4613**	539.0653
	KH-PSO	**45.56**	**0.0019**	0.0013	**46.04**	55.5356	501.9360
	K-Means	36.80	0.0000	0.0000	36.80	62.8879	464.7720
Balance Scale	K-FSS	51.52	**0.0025**	0.0017	51.76	**2.2762**	5.4314
	KH-FSS	51.41	0.0783	0.0518	**60.42**	2.7311	0.1409
	K-PSO	52.05	0.0099	0.0066	53.69	2.2768	5.4221
	KH-PSO	30.30	0.0415	0.0275	38.58	2.7521	0.0094
	K-Means	**53.69**	0.0000	0.0000	53.69	2.2768	**5.4499**
Sonar	K-FSS	53.33	**0.0216**	0.0143	56.52	2.6308	**6.7224**
	KH-FSS	**54.69**	0.0358	0.0237	**63.29**	2.9050	6.3293
	K-PSO	53.53	0.0378	0.0250	57.49	**1.1260**	2.5358
	KH-PSO	53.82	0.0448	0.0297	60.87	1.2826	0.0482
	K-Means	54.59	0.0000	0.0000	54.59	1.1309	2.4977

serial version of the four SCAs: K-FSS, KH-FSS, K-PSO and KH-PSO, and for the standard K-Means algorithm. Additionally, it is presented as internal quality criteria the within group variance and between group variance, which can be calculated by *sum-of-squares within cluster* (SSW) and *sum-of-squares between clusters* (SSB) [76], respectively, for the best run. Similarly, the results for the parallel versions are found in Table 5. The best results according to each measure were highlighted in black.

Comparing the results of clustering according to the fitness functions, we can conclude that KH-FSS produced better average accuracies in relation to K-FSS in the most cases for both serial and parallel versions. However, KH-PSO did not present great gains

Table 6
Speed-ups of the Swarm Clustering Algorithms.

Dataset	FSS				PSO			
	Algorithm	Serial time (s)	Parallel time (s)	Speed-up	Algorithm	Serial time (s)	Parallel time (s)	Speed-up
Iris	K-FSS	35.6928	1.**0655**	**33.4986**	K-PSO	**11.5486**	**3.0983**	**3.7458**
	KH-FSS	**35.0236**	1.0904	32.1199	KH-PSO	11.6141	3.5615	3.2610
Breast	K-FSS	227.2970	6.**9841**	**32.5449**	K-PSO	**81.4196**	**20.4813**	**3.9753**
Cancer	KH-FSS	**224.6856**	7.2009	31.2024	KH-PSO	80.7193	22.0505	3.6606
Glass	K-FSS	221.1663	4.**2167**	**52.4501**	K-PSO	**75.2373**	**16.4224**	**4.5813**
	KH-FSS	**217.6032**	4.2510	51.1887	KH-PSO	73.6492	17.4595	4.2182
Wine	K-FSS	138.3363	**3.0014**	**46.0905**	K-PSO	43.4506	**9.7437**	**4.4593**
	KH-FSS	**137.0587**	3.0124	45.4981	KH-PSO	**42.4273**	10.2539	4.1376
Olive	K-FSS	842.9974	**18.0180**	**46.7864**	K-PSO	260.7402	**57.8075**	**4.5104**
	KH-FSS	**790.8808**	18.0492	43.8180	KH-PSO	**255.0572**	62.0197	4.1125
Crude	K-FSS	16.9229	**0.6958**	**24.3215**	K-PSO	5.5209	**1.7097**	**3.2291**
Oil	KH-FSS	**16.3738**	0.7192	22.7666	KH-PSO	**5.4678**	1.7939	3.0479
Diabetes	K-FSS	57.8025	**1.3104**	**44.1105**	K-PSO	16.8261	**4.7688**	**3.5283**
	KH-FSS	**56.6468**	1.3245	42.7684	KH-PSO	**16.4594**	5.2073	3.1608
Egyptian	K-FSS	22.7011	**0.7925**	**28.6449**	K-PSO	7.5769	**2.0124**	**3.7651**
Skulls	KH-FSS	**20.9367**	0.8143	25.7112	KH-PSO	**6.9356**	2.2541	3.0768
Heart	K-FSS	135.8310	**3.2230**	**42.1442**	K-PSO	45.1653	**9.8342**	**4.5926**
Statlog	KH-FSS	**134.4223**	3.2308	41.6065	KH-PSO	**44.8955**	10.2336	4.3870
Ionosphere	K-FSS	**422.8466**	**11.4285**	36.9993	K-PSO	138.6299	**31.4948**	**4.4016**
	KH-FSS	424.8823	11.4473	**37.1163**	KH-PSO	**137.9243**	32.3201	4.2674
Vehicle	K-FSS	1159.3722	**31.3030.**	**37.0370**	K-PSO	360.8675	**78.5882**	**4.5918**
	KH-FSS	**1123.1130**	31.3404	35.8359	KH-PSO	**358.7570**	81.7535	4.3882
Balance	K-FSS	150.7235	**4.7549**	**31.6985**	K-PSO	49.1976	**11.7033**	**4.2037**
Scale	KH-FSS	**142.4017**	4.7782	29.8023	KH-PSO	**47.0105**	13.5953	3.4578
Sonar	K-FSS	**431.3376**	**14.9682**	**28.8169**	K-PSO	141.7698	**33.1670**	**4.2744**
	KH-FSS	433.2736	15.0993	28.6949	KH-PSO	**139.2036**	33.7617	4.1231

against K-PSO in both versions. For large search space, PSO presents premature convergence to local optima. It is known in the literature that PSO suffers with stagnation behavior more than others SI algorithms. This situation consists of not generating alternative solutions after a certain number of iterations. In the case of PSO, it is due to the weak fine tuning given by the balance between the global and local exploration abilities.

It was also observed that the FSS and PSO algorithms have good convergence, indicating very low standard deviation at 10 runs of the experiments. Only 4 of the 52 cases (13 datasets × 4 algorithms) presented significant differences ($P < 5\%$) in the serial implementations (K-PSO in Iris and Diabetes, KH-PSO in Breast Cancer and KH-FSS in Balance Scale), as expressed in Table 4. Concerning to the results of the parallel versions, just 3 cases (KH-FSS in Olive and Balance Scale and K-PSO in Diabetes) exhibited significant differences.

For most of the datasets (9 in 13) in Table 4, both serial K-SCA and KH-SCA algorithms outperformed the standard K-Means or are close to the best results produced by this algorithm in terms of average accuracy. The same results were found for the parallel algorithms in Table 5. Observing the best runs, the swarm algorithms overcame the standard K-Means in 12 datasets and for all datasets, respectively, in the serial and parallel versions.

K-Means achieved better accuracies than K-SCA and KH-SCA for Olive, Heart Statlog, Ionosphere and Balance Scale datasets. Observing Table 3, we note that Olive, Ionosphere and Balance Scale contain unbalanced clustering patterns, where the amount of samples is irregularly distributed in the dataset. On the other hand, Heart Statlog is the unique dataset formed by some different types of attributes, especially binary attributes. These remarks could indicate that SCA exposes deadlocks to tackle unbalanced data and binary variables. The SI algorithms applied in this work consider the optimization for continuous domain problems.

In the majority cases, it was also observed that K-FSS produced better accuracy than K-PSO in experiments in serial version, while K-PSO showed better than K-FSS in the parallel version. We explain this difference because the location updating in PSO is made using only both information about each particle and the best particle in the last iteration. So, in the parallel version, there is no difference for moving the particles through the space compared to serial version, because each particle movement is separated and independent. For FSS, the movement depends of direction and barycenter computed by information about all fishes. Thus, in serial version these parameters are the same for all fishes. Since in parallel version we have used asynchronous processing, each fish can have different computations of these parameters at time. In consequence, parallel version of FSS trends to have lower precision compared with the serial implementation, because the parameters are not updated equally (at the same time) for all fishes. Serial KH-FSS obtained better clustering than serial KH-PSO. Considering parallel K-Harmonic, there was no emphasis on any algorithm. FSS has a local search mechanism more efficient than PSO. So, FSS yielded better accuracy than PSO in the serial version.

Observing the variation difference between serial and parallel average accuracy presented in Tables 4 and 5 for all datasets, the average accuracy decreased from serial to parallel mode 2.29%, 1.94% and 1.09%, respectively, for K-FSS, KH-FSS and KH-PSO, while K-PSO average accuracy increased by 4.74%. As explained above, analogous reasoning of dependence in serial and parallel implementations is also related to the K-Means (independent of samples) and K-Harmonic (dependent of all samples) calculations.

In general, K-SCA algorithms presented lower SSW and higher SSB than KH-SCA and standard K-Means, as seen in Tables 4 and 5 for the best run. In the serial version K-FSS obtained the lowest SSW for 9 datasets. For the parallel version the best SSW performance occurred on 7 datasets with K-PSO. Taking into account the SSB, the highest score was reached in 8 datasets with K-PSO in the serial mode and in 7 datasets with K-FSS in the parallel mode. However, it was not related any direct correlation of SSW and SSB with the best found results for accuracy.

Considering dissimilarities measures, the experiments demonstrated that SSW is better minimized in K-FSS serial and K-PSO parallel. SSB is better maximized in K-PSO serial and K-FSS parallel.

Table 6 demonstrates the performance differences in terms of computing time between the CPU-based and GPU-based algorithms, as well the speedup of the parallel implementations against

the serial ones. The speed-up is calculated as the ratio of the serial runtime to the parallel time. The best results according to each measure were highlighted in black.

Through our experiments, we observed that K-FSS and KH-FSS clustering on the GPU has a runtime that is significantly smaller than its CPU counterpart, getting a speed-up from 22 to 52 times faster. Comparing K-FSS and KH-FSS time executions, we have noted that KH-FSS spent lower runtime than K-FSS in the serial experiments, but for the parallel implementation K-FSS overcame KH-FSS in smaller runtime. The same behavior was verified for PSO. They can be explained because KHM calculation is more complex than Euclidean distance (in K-Means), consuming larger time during exchange data from CPU to GPU. Thus, the speedup gain was greater to K-FSS than to KH-FSS. Analogously, the same was noted to K-PSO compared to KH-PSO.

Regarding to PSO-SCA, the parallel implementation can still achieve 3–4 times speed-up compared to the serial one. The gain is not so expressive like FSS because its original serial algorithm is already quite simplified. However, PSO-SCA serial version is much faster than the serial FSS-SCA algorithm.

The parallel FSS-SCA runtime was well below the parallel PSO-SCA runtime. This is explained by the fact that the procedures in CUDA require exchange data from CPU to GPU. In the case of PSO, because of its procedural simplicity, this exchange consumed most part of the computing time. The main bottleneck in the CUDA architecture occurs in the data transferring between the host (CPU) and the device (GPU). In short, in our experiments, in terms of runtime, it is more advantageous to use PSO-SCA in serial clustering and FSS-SCA in parallel clustering, regardless of fitness function.

Reducing the population size in swarm algorithms is required to prevent slow convergence on serial implementations. It is possible to use GPUs to increase the number of individuals in order to take better use of the resources, because in the parallel implementation the execution time suffers less variation in the computing volume, due to GPU capabilities to execute a large number of concurrent threads. Therefore, the search space for the clustering solution can be intensively explored with a huge population size in GPU without sacrificing performance.

8. Conclusions

This work investigated the application of two swarm algorithms (FSS and PSO) based on K-Means algorithm and on K-Harmonic Means to cluster analysis and they were compared to the standard K-Means. To evaluate the performance of the proposed algorithm we have tested it on 13 benchmark datasets with three internal cluster validity measures. We have also evaluated the performance of the SCAs implemented in serial mode running on CPU and in parallel mode running on GPU.

We have found that FSS performed better than PSO and standard K-Means in most experiments in serial mode for the two fitness functions. FSS was able to explore the clustering space better than PSO. In parallel mode, PSO was better than FSS, because of accuracy losses due to asynchronous fish's weight updating. The results support further evidence confirming that SI algorithms do advance the progression of reaching global optimum centroids and partitional clustering algorithms can be merged with SI algorithms to scape of local optima in Data Clustering.

Besides, the experiments also reveal that obtaining better fitness does not ensure higher accuracy. If the primordial goal is to elevate the accuracy, more related similarity measures to accuracy could be applied as fitness function.

The comparative study of the proposed SCA asserts the potentiality of the adopted approach. SCA procedure is independent of the SI algorithm. SCA is flexible and allows using any dissimilarity measure as fitness function for clustering. The higher accuracy obtained in the datasets clustering by PSO and FSS algorithms, confronted by standard K-Means, encourages to further studies.

We have observed a significant reduction of the computing execution time of the parallel SCAs when compared to the serial implementations, giving in the most cases similar quality results compared to the serial algorithm. FSS ran faster than PSO for the parallel implementation of the SCA with low accuracy's losses. Thus, swarm based algorithms are well suited for parallelization in the clustering task.

We conclude that the FSS is an efficient, reliable and robust method, which can be applied successfully to generate optimal cluster centers. However, for parallel processing it must be improved to reduce precision losses.

Regarding to PSO runtime, the speedup gain of standard PSO running on GPU was not greatly superior to serial processing in spite of its parallel nature, because of the time consuming for memory transactions between host and device. The most PSO variants require more complex computing, introducing extra communication and bottlenecks that are inefficient in a parallel environment. Some higher advantage of the parallel PSO algorithm could be considered just for huge datasets processing. However, a GPU-PSO scabilitity study must be conducted to confirm that.

For future works, we intend to evaluate others dissimilarity measures related to clustering validity criteria as fitness function for SCA with FSS. Larger datasets could be investigated as well. Further research includes evaluating the performance of parallel algorithms using GPU by varying the population size for large values. The SCA will also be extended to dynamically determine the optimal number of clusters. Different SI algorithms will be included for evaluating as well.

Acknowledgments

The authors would like to thank Fundação de Amparo à Pesquisa do Estado de São Paulo for the financial support (Process FAPESP N° 2013/08741-0, Process FAPESP N° 2013/08730-8 and Process FAPESP N° 2013/23027-1) and PROPE/UNESP (Process N° 2345/002/14-PROPe/CDC).

References

[1] A.R. Jain, M.N. Murthy, P.J. Flynn, Data clustering: a review, ACM Comput. Surv. 31 (3) (1999) 265–323.
[2] S.P. Lloyd, Least squares quantization in PCM, IEEE Trans. Inf. Theory 28 (2) (1982) 129–137.
[3] B. Zhang, M. Hsu, U. Dayal, K-Harmonic Means: A data clustering Algorithm. Technical Report HPL1999-124, Hewlett-Packard Labs, 1999.
[4] Y. Kao, K. Cheng, An ACO-based clustering algorithm, in: Proceedings of the 5th International Workshop on Ant Colony Optimization and Swarm Intelligence, 2006, pp. 340–347.
[5] M. Fathian, B. Amiri, A. Maroosi, Application of honey-bee mating optimization algorithm on clustering, Appl. Math. Comput. 190 (2) (2007) 1502–1513.
[6] Z. Güngör, A. Ünler, K-harmonic means data clustering with simulated annealing heuristic, Appl. Math. Comput. 184 (2) (2007) 199–209.
[7] B. Auffarth, Clustering by a genetic algorithm with biased mutation operator, in: Proceedings of the IEEE Congress Evolutionary Computation (CEC), 2010, pp. 1–8.
[8] Y. Liu, H. Fu, An effective clustering algorithm with ant colony, J. Comput. 5 (4) (2010) 598–605.
[9] M. Yaghini, N. Ghazanfari, Tabu-KM, a hybrid clustering algorithm based on Tabu search approach, Int. J. Ind. Eng. Prod. Res. 21 (2) (2010) 71–79.
[10] C.S. Zhang, D.T. Ouyang, J.X. Ning, An artificial bee colony approach for clustering, Expert Syst. Appl. 37 (2010) 4761–4767.
[11] S. Goel, A. Sharma, P. Bedi, Cuckoo search clustering algorithm: a novel strategy of biomimicry, in: Proceedings of the 2011 World Congress on Information and Communication Technologies (WICT), IEEE, 2011, pp. 916–921.
[12] Z. Huang, Y. Zhou, Using glowworm swarm optimization algorithm for clustering analysis, J. Converg. Inf. Technol. 6 (2) (2011) 78–85.
[13] D. Karaboga, C. Ozturk, A novel clustering approach: Artificial Bee Colony (ABC) algorithm, Appl. Soft Comput. 11 (2011) 652–657.
[14] W. Kuedlo, A clustering method combining differential evolution with the K-means algorithm, Pattern Recognit. Lett. 32 (12) (2011) 1613–1621.

[15] J. Senthilnath, S.N. Omkar, V. Mani, Clustering using firefly algorithm: performance study, Swarm Evol. Comput. 1 (3) (2011) 164–171.
[16] M. Yin, Y. Hu, F. Yang, X. Li, W. Gu, A novel hybrid K-harmonic means and gravitational search algorithm approach for clustering, Expert Syst. Appl. 38 (2011) 9319–9324.
[17] A.A. Abshouri, A. Bakhtiary, A new clustering method based on firefly and KHM, J. Commun. Comput. 9 (4) (2012) 387–391.
[18] T. Hassanzadeh, M.R. Meybodi, A new hybrid approach for data clustering using firefly algorithm and K-means, in: Proceedings of the CSI International Symposium on Artificial Intelligence and Signal Processing, IEEE Press, New York, 2012, pp. 7–11.
[19] A. Hatamlou, S. Abdullah, H. Nezamabadi-pour, A combined approach for clustering based on K-means and gravitational search algorithms, Swarm Evol. Comput. 6 (2012) 47–52.
[20] G. Liu, H. Huang, S. Wang, Z. Chen, A novel spatial clustering analysis method using bat algorithm, Int. J. Adv. Comput. Technol. 1 (20) (2012) 561–571.
[21] Y. Liu, X. Wu, Y. Shen, Cat swarm optimization clustering (KSACSOC): a cat swarm optimization clustering algorithm, Sci. Res. Essays 7 (49) (2012) 4176–4185.
[22] S.C. Tan, Simplifying and improving swarm-based clustering, in: Proceedings of 2012 IEEE Congress on the Evolutionary Computation (CEC), Brisbane, Australia, 2012, pp. 1–8.
[23] R. Tang, S. Fong, X.-S. Yang, S. Deb, Integrating natureinspired optimization algorithms to K-means clustering, in: Proceedings of the 7th International Conference on Digital Information Management (ICDIM '12), 2012, pp. 116–123.
[24] X. Yan, Y. Zhu, W. Zou, L. Wang, A new approach for data clustering using hybrid artificial bee colony algorithm, Neurocomputing 97 (2012) 241–250.
[25] A. Hatamlou, Black hole: a new heuristic optimization approach for data clustering, Inf. Sci. 222 (2013) 175–184.
[26] J. Senthilnath, V. Das, S.N. Omkar, V. Mani, Clustering using Lévy Flight Cuckoo Search, in: J.C. Bansal, P. Singh, K. Deep, M. Pant, A. Nagar (Eds.), Proceedings of Seventh International Conference on Bio-Inspired Computing: Theories and Applications, (BIC-TA 2012). AISC, vol. 202, 2012, pp. 65–75.
[27] H. Shah-Hosseini, Improving K-means clustering algorithm with the intelligent water drops (IWD) algorithm, Int. J. Data Min. Model. Manage. 5 (4) (2013) 301–317.
[28] P. Manikandan, S. Selvarajan, Data clustering using Cuckoo Search Algorithm (CSA), in: B.V. Babu, A. Nagar, K. Deep, M. Pant, J.C. Bansal, K. Ray, U. Gupta (Eds.), Proceedings of the Second International Conference on Soft Computing for Problem Solving (SocProS 2012). AISC, vol. 236, 2014, pp. 1275–1283.
[29] S. Fong, S. Deb, X.-S. Yang, Y. Zhuang, Towards enhancement of performance of K-Means clustering using nature-inspired optimization algorithms, Sc. World J. 2014 (2014), Article ID 564829, 16 pages.
[30] I.B. Saida, K. Nadjet, B. Omar, A new algorithm for data clustering based on Cuckoo Search Optimization, Genet. Evol. Comput. 238 (2014) 55–64.
[31] D.W. van der Merwe, A.P. Engelbrecht, Data clustering using particle swarm optimization, in: Proceedings of the 2003 IEEE Congress on Evolutionary Computation, IEEE Service Center, Piscataway, NJ, 2003, pp. 215–220.
[32] F. Ye, C.Y. Chen, Alternative KPSO-clustering algorithm, Tamkang J. Sci. Eng. 8 (2) (2005) 165–174.
[33] N.I. Ghali, N. El-Dessouki, A.N. Mervat, L. Bakrawi, Exponential particle swarm optimization approach for improving data clustering, Int. J. Comput. Electr. Autom. Control Inf. Eng. 2 (6) (2008) 1818–1822.
[34] K. Premalatha, A.M. Natarajan, A new approach for data clustering based on PSO with local search, Comput. Inf. Sci. 1 (4) (2008) 139–145.
[35] Y. Kao, S.Y. Lee, Combining K-means and particle swarm optimization for dynamic data clustering problems, in: Proceedings of the IEEE International Conference on Intellgent Computing and Intelligent System, 2009.
[36] F. Yang, T. Sun, C. Zhang, An efficient hybrid data clustering method based on K-harmonic means and Particle Swarm Optimization, Expert Syst. Appl. 36 (2009) 9847–9852.
[37] S. Rana, S. Jasola, R. Kumar, A hybrid sequential approach for data clustering using K-Means and particle swarm optimization algorithm, Int. J. Eng. Sci. Technol. 2 (6) (2010) 167–176.
[38] J. Chen, Hybrid clustering algorithm based on PSO with the multidimensional asynchronism and stochastic disturbance method, J. Theor. Appl. Inf. Technol. 46 (1) (2012) 434–440.
[39] L.-Y. Chuang, Y.-D. Lin, C.-H. Yang, An improved particle swarm optimization for data clustering, in: Proceedings of the International MultiConference of Engineers and Computer Scientists (IMECS), 2012.
[40] M. Neshat, S.F. Yazdi, D. Yazdani, M. Sargolzaei, A new cooperative algorithm based on PSO and K-Means for data clustering, J. Comput. Sci. 8 (2) (2012) 188–194.
[41] S. Rana, S. Jasola, R. Kumar, A boundary restricted adaptive particle swarm optimization for data clustering, Int. J. Mach. Learn. Cybern. (2012) 391–400.
[42] S. Satapathy, D. Maheshwari, S. Hanuman, V. Babu, P.K. Patra, B.N. Biswal, Integrated PSO and DE for data clustering, Int. J. Mach. Learn. Comput. 2 (6) (2012) 839–843.
[43] M. El-Tarabily, R. Abdel-Kader, M. Marie, G. Abdel-Azeem, A PSO-based subtractive data clustering algorithm, Int. J. Res. Comput. Sci. 3 (2) (2013) 1–9.
[44] R.J. Kuo, F.E. Zulvia, Automatic clustering using an improved particle swarm optimization, J. Ind. Intell. Inf. 1 (1) (2013) 46–51.
[45] S.C. Satapathy, A. Naik, Efficient clustering of dataset based on particle swarm optimization, Int. J. Comput. Sci. Eng. Inf. Technol. Res. 3 (1) (2013) 39–48.
[46] N. Singh, D. Singh, The improved K-Means with particle swarm optimization, J. Inf. Eng. Appl. 3 (11) (2013) 1–7.
[47] M. Lashkari, A. Rostami, Extended PSO algorithm for improvement problems K-Means clustering algorithm, Int. J. Manag. Inf. Technol. 6 (3) (2014) 17–29.
[48] G. Saini, H. Kaur, A novel approach towards K-Mean clustering algorithm with PSO, Int. J. Comput. Sci. Inf. Technol. 5 (4) (2014) 5978–5986.
[49] X. Yang, P. Liu, Tayloring fuzzy C-Means clustering for big data using random sampling and particle swarm optimization, Int. J. Database Theory. Appl. 8 (3) (2015) 191–202.
[50] Y. Zheng, Y. Zhou, J. Qu, An improved PSO clustering algorithm with entropy-based fuzzy clustering, WSEAS Trans. Comput. 14 (2015) 88–96.
[51] A. Abraham, S. Das, S. Roy, Swarm intelligence algorithms for data clustering, in: Soft Computing for Knowledge Discovery and Data Mining, 2008, pp. 279–313.
[52] NVIDIA Inc., NVIDIA CUDA: Programming Guide. Version 6.5, 2014.
[53] J. Li, X. Hu, Z. Pang, K. Qian, A parallel ant colony optimization algorithm based on fine-grained model with GPU-acceleration, Int. J. Innov. Comput. Inf. Control 5 (2009) 3707–3716.
[54] J.M. Cecilia, J.M. García, M. Ujaldon, A. Nisbet, M. Amos, Parallelization strategies for ant colony optimisation on GPUs, in: Proceedings of the 2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011, pp. 339–346.
[55] L. Mussi, Y.S.G. Nashed, S. Cagnoni, GPU-based asynchronous particle swarm optimization, in: Proceedings of the 13th Annual Genetic and Evolutionary Computation Conference GECCO '11, 2011, pp. 1555–1562.
[56] D.L. Souza, G.D. Monteiro, T.C. Martins, V.A. Dmitriev, O.N. Teixeira, PSO-GPU: accelerating particle swarm optimization in CUDA-based graphics processing units, in: Proceedings of the 13th Annual Genetic and Evolutionary Computation Conference (GECCO '11), 2011, pp. 837–838.
[57] J.M. Cecilia, A. Nisbet, M. Amos, J.M. García, M. Ujaldon, Enhancing GPU parallelism in nature-inspired algorithms, J. Supercomput. 63 (3) (2013) 773–789.
[58] L. Dawson, I. Stewart, Candidate set parallelization strategies for ant colony optimization on the GPU, in: Di Martino Beniamino, Talia Domenico, Xiong Kaiqi (Eds.), Algorithms and Architectures for Parallel Processing Joanna Kolodziej, Lecture Notes in Computer Science, vol. 8285, Springer International Publishing, 2013, pp. 216–225.
[59] K. Ding, S. Zheng, Y. Tan, A GPU-based parallel fireworks algorithm for optimization, in: Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation, GECCO '13, New York, USA, 2013, pp. 9–16.
[60] R. Jovanovic, M. Tuba, Parallelization of the Cuckoo Search using CUDA architecture, in: Proceedings of the 7th International Conference on Applied Mathematics, Simulation, Modelling (ASM '13), 2013.
[61] P. Krömer, J. Platos, V. Snasel, Nature-inspired meta-heuristics on modern GPUs: state of the art and brief survey of selected algorithms, Int. J. Parallel Program. 42 (5) (2014) 681–709.
[62] G. Luo, S. Huang, Y. Chang, S. Yuan, A parallel bees algorithm implementation on GPU, J. Syst. Architect. 60 (3) (2014) 271–279.
[63] W. Situ, Y.-K. Lam, Y. Xiao, P.W.M. Tsang, C.-S. Leung, GPU accelerated PK-means algorithm for gene clustering, in: Proceedings of the Proceedings of the 2011 International Conference on Bioinformatics & Computational Biology, 2011.
[64] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of IEEE International Conference on Neural Networks, 1995, pp. 1942–1948.
[65] C.J.A. Bastos-Filho, F.B. Lima Neto, A.J.C.C. Lins, A.I.S. Nascimento, M.P. Lima, A novel search algorithm based on fish school behavior, in: Proceedings of the 2008 IEEE International Conference on Systems, Man, and Cybernetics, 2008, pp. 2646–2651.
[66] C.J.A. Bastos-Filho, F.B. Lima-Neto, M.F.C. Sousa, M.R. Pontes, S.S. Madeiro, On the influence of the swimming operators in the fish school search algorithm, in: Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, 2009, pp. 5012–5017.
[67] A.J.C.C. Lins, C.J.A. Bastos-Filho, D.N.O. Nascimento, M.A.C. Oliveira Júnior, F.B. Lima Neto, Analysis of the performance of the fish school search algorithm running in graphic processing units, in: Parpinelli Rafael (Ed.), Theory and New Applications of Swarm Intelligence, InTech, 2012, pp. 17–32.
[68] C.J.A. Bastos-Filho, D.O. Nascimento, An enhanced fish school search algorithm, in: Proceedings of the 1st BRICS Countries Congress (BRICS-CCI) and 11th Brazilian Congress (CBIC) on Computational Intelligence, 2013, pp. 1–6.
[69] S. Rana, S. Jasola, R. Kumar, A review on particle swarm optimization algorithms and their applications to data clustering, Artif. Intell. Rev. 35 (2011) 211–222.
[70] A.A.A. Esmin, R.A. Coelho, S. Matwin, A Review on Particle Swarm Optimization Algorithm and Its Variants to Clustering High-Dimensional Data, Springer, 2013, pp. 1–23.
[71] S. Sarkar, A. Roy, B.S. Purkayastha, Application of particle swarm optimization in data clustering: a survey, Int. J. Comput. Appl. 65 (25) (2013) 38–46.
[72] S. Alam, G. Dobbie, Y.S. Koh, P. Riddle, S. Ur Rehman, Research on particle swarm optimization based clustering: a systematic review of literature and techniques, Swarm Evol. Comput. 17 (2014) 1–13.
[73] A.A.A. Esmin, R.A. Coelho, S. Matwin, A review on particle swarm optimization algorithm and its variants to clustering high-dimensional data, Artif. Intell. Rev. 44 (2015) 23–45.
[74] NVIDIA Inc., Toolkit 6.0 CURAND Guide, 2014, February.
[75] C.L. Blake, C.J. Merz, UCI Repository of Machine Learning Database, 2013, Available at: http://archive.ics.uci.edu/ml/datasets.html.
[76] N.X. Vinh, J. Epps, J. Bailey, Information theoretic measures for clusterings comparison: is a correction for chance necessary? in: Proceedings of the 26th International Conference on Machine Learning (ICML '09), 2009.