Clustering Based on Sequential Multi-Objective Games

Imen Heloulou[1], Mohammed Said Radjef[1], and Mohand Tahar Kechadi[2]

[1] University A. MIRA of Bejaia, Road Targua Ouzemour 06000, Bejaia, Algeria
[2] School of Computer Science and Informatics, University College Dublin (UCD), Belfield, Dublin 4, Ireland
heloulou.imen@gmail.com

Abstract. We propose a novel approach for data clustering based on sequential multi-objective multi-act games (ClusSMOG). It automatically determines the number of clusters and optimises simultaneously the inertia and the connectivity objectives. The approach consists of three structured steps. The first step identifies initial clusters and calculates a set of conflict-clusters. In the second step, for each conflict-cluster, we construct a sequence of multi-objective multi-act sequential two-player games. In the third step, we develop a sequential two-player game between each cluster representative and its nearest neighbour. For each game, payoff functions corresponding to the objectives were defined. We use a backward induction method to calculate Nash equilibrium for each game. Experimental results confirm the effectiveness of the proposed approach over state-of-the-art clustering algorithms.

Keywords: clustering, multi-objective, sequential game, multi-act, inertia, payoff functions, connectivity, backward induction, Nash equilibrium.

1 Introduction

Nowadays, data clustering is a well-established field, which is growing rapidly in many domains such as pattern-analysis and grouping, and decision-making [1]. Many clustering methods have been proposed to satisfy these application requirements. However, in many real-world problems, more than one objective are needed to be optimised. The multi-objective clustering methods attempt to identify clusters in such a manner that several objectives are optimised during the procedure [1]. Traditionally, multi-objective methods have been categorised as ensemble, evolutionary and micro-economic methods. Cluster ensemble frameworks combine different partitions of data using consensus functions [2]. Ensemble methods based clustering have been proven to more powerful method than individual clustering methods. However, they are not able to deal with multi-objective optimisation [3]. On the other hand, evolutionary algorithms such as MOEA [4], PESA-II [3] and MOCK [4] identify better clusters than ensemble clustering methods, as they optimise many objectives concurrently [5]. Microeconomic models naturally analyse the situations of conflicting objectives in a game theoretic setting [6].

Gupta et al [7] and Badami et al [8] used a microeconomic game theoretic approach for clustering, which simultaneously optimises compaction and equi-partitioning. Garg et al [9] proposed the use of Shapley value to give a good start to K-means. Bulo and Pelillo [10] used the concept of evolutionary games for hyper-graph clustering.

Despite the large number of algorithms, the priority is always given to partitioning algorithms. They are attractive as they lead to elegant mathematical and algorithmic proofs and settings. However, there are several limitations with this oversimplified formulation. Probably the best-known limitation of the partitioning approaches is the typical requirement of the number of clusters to be known in advance, which is not always easy, especially when there is no sufficient information on the data set. The choice of the clusters' centres represents another major problem. Moreover, the partitioning techniques were only good in minimising the criterion of compactness and in detecting clusters of spherical form. In our endeavour to provide answers to the questions raised above, we found that game theory offers a very elegant and general perspective that serve well our purposes and which has found applications in diverse fields. Specifically, in this paper we have developed a clustering technique based on non-cooperative games theory in sequential form. This novel approach performs the optimisation on the basis of two conflicting objectives, inertia and connectivity in a simultaneous manner. We use a backward induction method to derive the right number of the clusters. In this way, not only the number of clusters is determined dynamically, but also the distribution of objects to clusters will be also done by negotiation.

2 Multi-Objective Clustering Game

Before describing our clustering approach, lets define the fundamentals of every clustering algorithm, which is the similarity measure. The similarity measure allows us to evaluate how much clustering is good or bad by evaluating either intra-cluster or inter-cluster inertia or both. In our case we attempt to optimise intra-cluster/inter-cluster inertia and the connectivity objectives. The intra-cluster inertia should be as small as possible in order to have a set of homogeneous clusters. Let ξ a set of clusters of the initial data set $\boldsymbol{D}$. The inter-cluster inertia is given by [11]:

$$I_A(\xi) = \frac{1}{n}\sum_{C_i\in\xi}\sum_{j\in C_i} d(j, ch_i) \tag{1}$$

where $\boldsymbol{d}$ is the Euclidean distance between object $\boldsymbol{j}$ and the centre $\boldsymbol{ch_i}$ of the cluster $\boldsymbol{C_i}$. A larger value of the inter-cluster inertia leads to a good separation of the clusters. It can be calculated using the following relationship [11]:

$$I_R(\xi) = \frac{1}{n}\sum_{i=1}^{K}|C_i|\ d(ch_i, g) \tag{2}$$

where $\boldsymbol{g = (g_1, \ldots, g_j, \ldots, g_\delta)}$ and $\boldsymbol{g_j}$ is the gravity centre of $\boldsymbol{D}$ along the jth dimension.

$$g_j = \frac{1}{n}\sum_{i=1}^{n} i_j \tag{3}$$

The connectivity measure evaluates which neighbouring data objects are placed in the same cluster. It is computed as follows [12]:

$$Connc(\xi) = \frac{1}{n}\sum_{i=1}^{n} \frac{\sum_{j=1}^{L} x_{i,nn_{ij}}}{L} \tag{4}$$

where:

$$x_{rs} = \begin{cases} 1 \; if \; \exists \, C_l : r, s \in C_l \\ 0, \quad otherwise \end{cases}$$

nn**$_{ij}$* is the j^{th} nearest neighbour of object ***i and ***L***, a parameter, is the number of neighbours that contribute to the connectivity measure.

Table 1. Notations and Terminology.

D	Data set; $D = \{1, 2, \dots i, \dots, n\}$ each object i is described by a set of δ attributes $i = (i_1, i_2, \dots, i_\delta)^{\mathrm{T}} \in \Re^\delta$
$C_i^{(t)}$	Cluster C_i at time *t*
$K^{(t)}$	Total number of clusters at time *t*
$\xi^{(t)}$	Set of clusters at time *t*; $\xi^{(t)} = \{C_1^{(t)}, \dots, C_{K^{(t)}}^{(t)} (K^{(t)} \leq n)\}$
ch_i	Center of cluster $C_i^{(t)}$; $Argmin_{m \in C_i^{(t)}} \frac{1}{\lvert C_i^{(t)} \rvert} \sum_{j \in C_i^{(t)}} d(m, j)$
h_D	*h* measurement calculates the dissimilarity of object i with respect to all dataset's objects; $h_D(i) = \frac{1}{n} \sum_{\substack{m \in D \\ i \neq m}} d(i, m)$
$I_A(C_i^{(t)})$	Intra-cluster inertia for cluster C_i at time *t*; $I_A(C_i^{(t)}) = \sum_{j \in C_i^{(t)}} d(j, ch_i)$
$Connc(C_i^{(t)})$	Connectivity of cluster C_i at time *t*; $Connc(C_i^{(t)}) = \frac{1}{\lvert C_i^{(t)} \rvert} \sum_{i=1}^{\lvert C_i^{(t)} \rvert} \frac{\sum_{j=1}^{L} x_{i,nn_{ij}}}{L}$
$neig(i)$	Set of *L* nearest neighbours of object i arranged in ascending order according to theirs Euclidean distance
ALC	Average linkage clustering measurement; $\frac{1}{\lvert C_i^{(t)} \rvert * \lvert C_m^{(t)} \rvert} \sum_{l \in C_i^{(t)}} \sum_{h \in C_m^{(t)}} d(l, h)$
$neig(C_i^{(t)})$	Nearest neighbour of cluster $C_i^{(t)}$ according to *ALC*; $neig(C_i^{(t)}) = Argmin_{C_m^{(t)} \in \xi^{(t)} \setminus C_i^{(t)}} \frac{1}{\lvert C_i^{(t)} \rvert * \lvert C_m^{(t)} \rvert} \sum_{l \in C_i^{(t)}} \sum_{j \in C_m^{(t)}} d(l, j)$
$E(i)$	Set of i's neighbours belonging to nearest neighbour of cluster $C_j^{(t)}$; $E(i) = neig(i) \cap neig\left(C_j^{(t)}\right)$, $\forall \, i \in C_j^{(t)}$
$E(C_j^{(t)})$	Set of all $E(i)$, $\forall \, i \in C_j^{(t)}$; $E(C_j^{(t)}) = \bigcup_{i \in C_j^{(t)}} E(i)$
$E(C_i^{(t)}, C_j^{(t)})$	Set of all neighbours in conflict between ch_i and ch_j; $E(C_i^{(t)}) \cap E(C_j^{(t)})$
$E_i(C_i^{(t)}, C_j^{(t)})$	For all $m \in E(C_i^{(t)}, C_j^{(t)})$, assuming: $C_i^{(t)} = C_i^{(t)} \cup \{m\}$ and calculating $d(m, ch_i)$. Then, ch_i constructs $E_i(C_i^{(t)}, C_j^{(t)})$ containing elements of $E\left(C_i^{(t)}, C_j^{(t)}\right)$ arranged in ascending order based to Euclidean distance according to ch_i
$B(C_i^{(t)}, C_j^{(t)})$	Set of objects which ch_i would to exchange with ch_j arranged in ascending order according to theirs Euclidean distance compared to ch_j ; $B\left(C_i^{(t)}, C_j^{(t)}\right) = \{m \in C_i^{(t)} / d(ch_i, m) > d\left(ch_j, m\right) \; and \; C_j^{(t)} = neig(C_i^{(t)})\}$
$conflict^{(t)}$	Set of conflict-clusters; $C_i^{(t)} \in conflict^{(t)} \Leftrightarrow \exists m, j \in \{1, \dots, K^{(t)}\} / \; C_m^{(t)}, C_j^{(t)} \in \xi^{(t)} \; and \; C_i^{(t)} = neig(C_j^{(t)}) = neig(C_m^{(t)})$
$conflict(C_i^{(t)})$	Set of clusters which for them $C_i^{(t)}$ is the nearest neighbour; $conflict\left(C_i^{(t)}\right) = \{C \in \xi^{(t)} / C_i^{(t)} = neig(C)\}$

R-Square (R^2) is used to estimate the number of clusters. It is defined by [13]:

$$R^2(\xi) = \frac{I_R(\xi)}{I_A(\xi) + I_R(\xi)} \tag{5}$$

The more close to 1 it is, the better is the classification. It should not be maximised at all costs, since it would lead thus to very large number of clusters [13]. By combining the connectivity and R^2 objectives, a trade-off is required to determine the appropriate value of ***K***. We expect this product to be large:

$$\varphi(\xi) = R^2(\xi) * Connc(\xi) \tag{6}$$

When we go beyond the right number of clusters, φ will decrease: the decrease in R^2 will be less significant but comes at a high cost in terms of connectivity (because a true cluster is being split). ClusSMOG consists of three components briefly explained in the following subsections. The notations and terminology used in the rest of the paper are given in Table 1.

2.1 Step 1: Initialisation

This step consists of initialising the primary $K^{(0)}$ clusters with their cluster representatives. Initially, we construct for each object $\boldsymbol{i}$ its ***neig(i)*** set and the similarity matrix. Then, we calculate for each object $\boldsymbol{i}$ the ***h_D(i)*** value. Objects with a minimum value of h_D means that they have high density around them and we could consider them as initial cluster representatives. A cluster representative should not be among the $5\%\ L$ first nearest neighbours of clusters representatives previously selected. Thereafter, the remaining objects are assigned to the nearest clusters according to d and then the cluster representatives are updated. After, each $\boldsymbol{C_i^{(0)}}$ seeks its nearest neighbour ***neig($C_i^{(0)}$)***, which is shown like resource to receive neighbouring objects. Due to this, a cluster may be the nearest neighbour of several clusters; it is called conflict-cluster, so the $conflict^{(0)}$ set is constructed containing conflict-clusters.

2.2 Step 2: Sequential Games for Conflict-Clusters Objects

The purpose of this step is to maximise φ value in order to achieve the correct number of clusters. At instant t, each $C_i^{(t)}$ seeks its nearest neighbour $neig(C_i^{(t)})$, which contains the neighbours of its objects, to integrate them in its cluster in order to increase its connectivity and maximise R^2. An issue may arise when a cluster is the nearest neighbour for several clusters, which at the same time trying to attract objects of this cluster, (***conflict-cluster***). However, the real competition will be on the objects coveted by different clusters. To analyse and find a solution to this competition, we have modelled it as a multi-objective multi-act sequential non-zero-sum game with perfect information. A game consists of a set of players, a set of moves (or strategies) available to those players, and a specification of payoffs for each combination of strategies. Sequential games are games where later players have some knowledge about earlier actions. The game has *perfect information* if each player, when making any decision, is perfectly informed of all the events that have previously occurred. Consequently, a problem may occur when conflict-cluster allocates all its objects. This cluster will be removed if its deletion improves the global objectives simultaneously. This step is, therefore, responsible for reducing the number of clusters.
The step starts by constructing the $conflict^{(t)}$ set, which contains all conflict-clusters. We define for each $C_l^{(t)} \in conflict^{(t)}$ the $conflict(C_l^{(t)})$ set of clusters for which $C_l^{(t)}$ is their nearest neighbour. The $conflict^{(t)}$ set is arranged in descending order according to the clusters cardinality, $\left|conflict\left(C_l^{(t)}\right)\right|$ and the elements of each $conflict(C_l^{(t)})$ set are ordered in ascending order according to *ALC*.

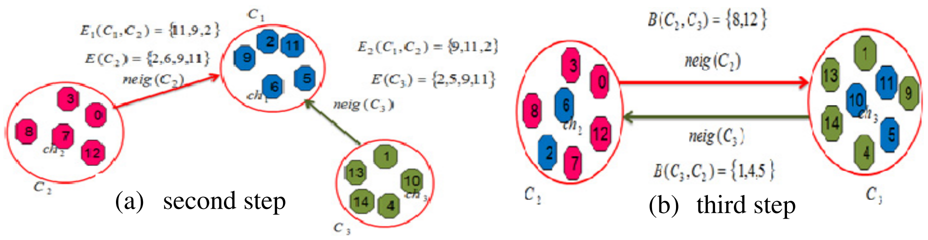


Fig. 1. An example of a clustering game

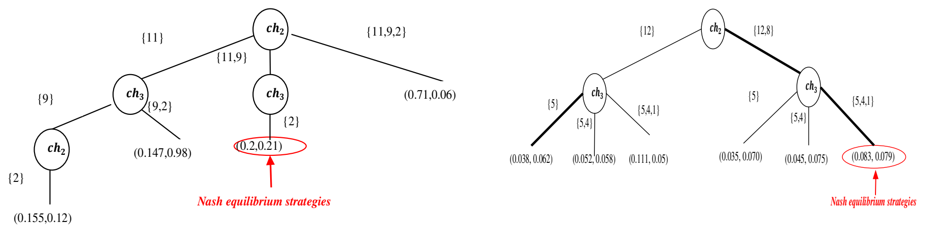
Starting by $C_l^{(t)} \in conflict^{(t)}$, which has the highest cardinality, a game is formulated between ch_i and ch_j, the representatives of both nearest clusters of $C_l^{(t)}$. So, the set of players competing for $C_l^{(t)}$ objects at time t is defined as follow:

$$\mathrm{N}^{(t)} = \{ch_i, ch_j / neig(C_i^{(t)}) = neig\left(C_j^{(t)}\right) = C_l^{(t)}\} \tag{7}$$

As shown in Figure 1a, a game is initiated between ch_2 and ch_3, where their clusters have the same neighbour, C_1. They try to attract their neighbouring objects in order to maximise their payoffs. To do this, each player constructs its $E(C_i^{(t)})$ set, which contains its neighbours belonging to the conflict-cluster. Each ch_i begins by integrating the objects that are not coveted by the opponent player; $E(C_i^{(t)}) \setminus E(C_i^{(t)}, C_j^{(t)})$ set. They affect objects that best improve the overall objectives, i.e. the φ value. However, the major issue that needs to be solved is when the objects are covered by both clusters, $E(C_i^{(t)}, C_j^{(t)})$. For example, object 11 is in conflict set because it is a neighbour of object $10 \in C_3$ and a neighbour of object $12 \in C_2$. The solution is to first affect the object to the player with higher connectivity degree; $Connc(C_i)^{(t)} > Connc(C_j)^{(t)}$. Thereafter, each ch_i arranges the elements of $E(C_i^{(t)}, C_j^{(t)})$ set in ascending order according to their distance to ch_i to construct the $E_i(C_i^{(t)}, C_j^{(t)})$ set. Sequential game is presented as tree (as shown in Figure 2). Each node represents a choice for a player. The links represent a possible action for that player. The payoffs are specified at the bottom of the tree. ch_2 moves first and chooses either to integrate in its cluster {11}, {11,9} or {11,9,2} objects. ch_3 sees ch_2's move and then chooses its action. It has the possibility to choose among remaining objects. Then, ch_2 is called again to choose its action. The same process is repeated until the end of conflict-objects (see Figure 2a). This game is called multi-act game; a player is allowed to act more than once. So, the actions set for a player ch_i at time t and the level p is defined as follows:

$$\mathrm{X}_i^{(t,p)} = \{M_l, l = 1,.., |E_i\left(C_i^{(t)}, C_j^{(t)}\right) \setminus (\cup^{\,p} < p\ (\mathrm{X}_i^{(t,p)} \cup \mathrm{X}_j^{(t,p)}))|\} \tag{8}$$

$$M_l = \{m_r \in E_i(C_i^{(t)}, C_j^{(t)}) \setminus (\cup^{\,p} < p\ (\mathrm{X}_i^{(t,p)} \cup \mathrm{X}_j^{(t,p)})), r \leq l\} \tag{9}$$



(a) Two-act sequential non-zero-sum two-player game

(b) Single-act sequential game non-zero-sum two-player game

Fig. 2. Sequential game Models

r is the rank of the object m in $E_i(C_i^{(t)}, C_j^{(t)})\setminus(\cup\ \ p < p\ (X_i^{(t,p)} \cup X_j^{(t,p)}))$. This allows ch_i to assign neighbours that maximise its payoff; provided that the chosen objects are not selected by any of the two players in the previous steps. We do not take all the possible objects coalitions but only the closest ones, as illustrated in Figure 2a. This definition reduces the complexity of the game.

The game performance is extremely restricted to proper definition of the payoffs function. The players choose their actions in order to maximise the connectivity of their clusters and maximise R^2. So the connectivity objective can be seen as a private objective and R^2 as a public objective (collective). So each cluster representative has a vector function (bi-criteria) of the payoffs:

$$f_i^{(t)}(x_1, x_2) = \left(Connc\left(C_i^{(t)}\right), R^2(\xi^{(t)})\right) \tag{10}$$

Where $f_i^{(t)}(.,.): X_1^{(t)} * X_2^{(t)} \xrightarrow{yields} \mathbb{R}^2$. The two objectives are conflicting, because connectivity's improvement can lead to the decrease of R^2 by reducing the number of the clusters. Every time we get rid of a conflict-cluster we increase the intra-cluster inertia. Decreasing R^2 will be less significant as each player competes for objects (resources) to improve their connectivity by trying to avoid a big loss in intra-cluster inertia.

After, we analyse the game using backward induction methodology to calculate the Nash equilibrium strategies, which represent the best structure of clustered data. A temporary reallocation of objects is performed according to the chosen actions. If the reallocations improve the overall objective according to φ, the allocations are committed, the clusters representatives are then updated and that conflict-cluster is removed. If the played game did not improve the system's objectives, a sequence of sequential games is formed for this conflict-cluster between all possible pairs of players starting by the nearest clusters. We have so a maximum $\sum_{C_i^{(t)} \in conflict^{(t)}} \frac{|conflict(C_i^{(t)})| * (conflict(C_i^{(t)}) - 1)}{2}$ games at time t. While $conflict^{(t)}$ set is not empty, a game is formulated for another conflict-cluster between two clusters representatives if theirs clusters are not changed in previous steps. If the $conflict^{(t)}$ set is empty and no improvement for the system thus go to third step, else restart again step2.

2.3 Step 3: Sequential Games for Clusters Neighbours

This step deals with intra-cluster inertia optimisation, i.e. construct homogeneous clusters. Each ch_i engages in exchanging objects with ch_j of its nearest neighbour cluster. This exchange between the two clusters is modelled as a single-act sequential non-zero-sum two-player game with perfect information, which we will identify its factors below. In a single-act game, each player makes a decision only once (as shown in Figure 2.b). The difference between this step and the previous step lies in the existence of common objects between the players, so the necessity to use multi-act in order to explore the effect of possible combinations of objects. Unlike Step 2, the set of exchanged objects are distinct. The players' set at time t is given by:

$$\mathrm{N}^{(t)} = \{ch_i, ch_j / C_j^{(t)} = neig(C_i^{(t)})\} \tag{11}$$

Before starting the game, each ch_i constructs the $B(C_i^{(t)}, C_j^{(t)})$ set. This set consists of objects that maximise its intra-cluster inertia and they will be ordered in ascending order according to their distance to the opponent player ch_j, since they will be transferred to him. The objects concerned are whose having minimal distance compared to the second player. As shown in Figure 1b, a game is formulated between ch_2 and ch_3, where ch_2 wants to exchange with ch_3 the $\{8,12\}$ objects because they are close to $ch_3 =$ object 10 rather than $ch_2 =$ object 6. This is the same for second player. Thus, the actions set for each player at time t is as follow:

$$\mathrm{X}_i^{(t)} = \{M_l, l = 1, \dots, |B(C_i^{(t)}, C_j^{(t)})|\} \tag{12}$$

$$M_l = \{m_r \in B(C_i^{(t)}, C_j^{(t)}), r \leq l\} \tag{13}$$

r is the rank of the object m in $B\left(C_i^{(t)}, C_j^{(t)}\right)$. The player having the smallest number of objects to exchange with the second player; $Argmin_{i,j}\left\{\left|B\left(C_i^{(t)}, C_j^{(t)}\right)\right|, \left|B\left(C_j^{(t)}, C_i^{(t)}\right)\right|\right\}$ is called leader and will play first.

It remains to define the payoffs function of the players. As we are interested in intra-cluster inertia; weaker intra-cluster inertia better is the homogeneity of objects, the payoffs function of each player is given by:

$$f_i^{(t)}(x_1, x_2) = \frac{1}{I_{A(C_i^{(t)})}} \tag{14}$$

Now, all elements characterising the game are well defined, so we can construct the tree representing the sequential form, as illustrated in Figure. 2b. To minimise the complexity of the tree, the number of combinations is reduced according to the order of objects in $B(C_i^{(t)}, C_j^{(t)})$, as shown in Figure. 2b. If a cluster is singleton, its object will be assigned immediately to the second player without playing the game.

After game's resolution by application of the backward induction, temporary reallocation of objects is performed according to the Nash equilibrium strategies representing the optimal intra-cluster inertia for both players. If the reallocations improve the overall objective; minimise the system‘s intra-cluster inertia $I_A(\xi^{(t)})$, the allocations are committed and the clusters representatives are then updated. This process is repeated at maximum $K^{(t)}$ times if the concerned clusters are not changed in the previous steps, because the neighbour of each cluster may change if the content of cluster is changed.

2.4 Solution Concept: Backward Induction

The backward induction is the most common solution concept for sequential games. Taking the example described at Fig. 2b, player 1 makes the first decision x_1 from its actions set $X_1 = \{\{8\},\{8,12\}\}$ and player 2 makes its decision x_2 from its actions set $X_2 = \{\{1\},\{1,4\},\{1,4,5\}\}$ after player 1. The payoffs functions of both players are given by $f_1(x_1,x_2)$ and $f_2(x_1,x_2)$ respectively. Backward induction uses the assumption of rationality, meaning that player 2 will maximise its payoff in any given situation. Player 2 chooses its best response x_2^* to the actions of player 1 which is the solution of this program:

$$x_2^* = Argmax_{x_2 \in X_2} f_2(x_1,x_2) \tag{15}$$

By anticipating the reaction x_2^* of player 2, we can reduce the size of our tree by eliminating the choices that player 2 will not choose. In this way, the links that maximise the player's payoff at the given information set are in bold. After this reduction, player 1 can maximise its payoffs once the player 2 choices are known. His best response is the solution of this program:

$$x_1^* = Argmax_{x_1 \in X_1} f_1(x_1,x_2^*) \tag{16}$$

The result (x_1^*,x_2^*) is a Nash equilibrium found by backward induction of player 1 choosing {12,8} and player 2 choosing {5,4,1}.

2.5 Stopping Criterion

If the overall objectives of the system are improved in the antecedent steps, this process starts again (step 2 and step 3) until no further improvement is possible.

3 Experimental Setup

We carried out extensive experimentations to compare ClusSMOG with state-of-the-art algorithms on several artificial datasets (Square1, Square4 [14], Ellipse [15], Dataset_9_2, Dataset_3_2, Dataset_4_3 [16]) and real-life datasets [17]. All experiments are implemented in Java and run on 2.20 GHz Intel core 2 Duo CPU with 3 GB RAM. Single-objective algorithms are performed using Rapid Miner [18] tool.

3.1 Parameters Settings

In this subsection, we discuss the specification of parameters for ClusSMOG and MOCK algorithms. For ClusSMOG, the initial number of clusters $\boldsymbol{K^{(0)}} = \sqrt{\boldsymbol{n}}$. While the choice of a reasonably large value of $\boldsymbol{L}$ is necessary to prevent outliers from being classified as individual clusters. However, if $\boldsymbol{L}$ is too large this may result in large number of small clusters. In our experiments, we chose $\boldsymbol{L = 10\%n}$, which allows robust detection of clusters and better performance on all studied datasets. MOCK was performed with the source codes available from [14].

3.2 Results and Discussion

In order to evaluate the performance of our algorithm, we have compared ClusSMOG to both single and multi-objective algorithms: MOCK, K-means, K-medoid, Dbscan and X-means based on different evaluation measure. Specifically we choose for internal measures the Purity, the Rand Index [19], the adjusted Rand index (ARI) [20] and the F-measure [21] and the Silhouette Index for external measures [22]. The results are summarised in Table 2 and show the robustness of ClusSMOG from the simultaneous optimisation of system's objectives: intra-cluster/inter-cluster inertia and connectivity. The best entries have been marked in bold in each row. While it may be marginally beaten by MOCK algorithm on Iris and Wine datasets and X-means on Suqare1 and Suqare4 datasets, it shows an impressive performance across the entire datasets. This is not only reflected in the high values of the Adjusted Rand Index, but also in the close agreement between the number of clusters in the generated solution, and the correct K on all datasets. It is clear that ClusSMOG and MOCK exceed X-means in the detection of the adequate number of clusters. ClusSMOG and X-means outperform MOCK on the Adjusted Rand Index value on Square4, Square1, Dataset_3_2 and Dataset_4_3 datasets. Our algorithm is better than MOCK and X-means on Glass, Dataset_9_2 and Ellipse datasets. In conclusion, our algorithm is able to detect clusters of arbitrary shapes; this is due to the simultaneous optimisation of connectivity and inertia objectives.

Table 2. Adjusted Rand Index value comparison between ClusSMOG, MOCK and X-means

Dataset				**ClusSMOG**		**MOCK**		**X-means**	
Name	**K**	***n***	**δ**	**K**	**ARI**	**K**	**ARI**	**K**	**ARI**
Square1	4	1000	2	4	0.963	4.22	0.9622	4	**0.9735**
Square4	4	1000	2	4	0.8274	4.32	0.7729	4	**0.8348**
Ellipse	2	400	2	5	**0.3017**	7.8	0.2357	4	0.2263
Dtaset_9_2	9	900	2	9	**0.8233**	8.52	0.8109	4	0.3353
Dtaset_3_2	3	76	2	3	**1**	3.33	0.9465	3	**1**
Dataset_4_3	4	400	3	4	**1**	3.78	0.8787	4	**1**
Iris	3	150	4	3	0.5962	3.05	**0.7287**	4	0.6744
Glass	6	214	9	6	**0.2449**	6.18	0.1677	4	0.2391
Wine	3	178	13	3	0.4331	3.59	**0.8647**	4	0.3034

We also evaluated and compared ClusSMOG with several others state-of-the-art algorithms like K-means, Dbscan and K-medoid using other internal measures evaluation. Figure. 3 shows the results of the value of purity on several datasets. ClusSMOG gives higher values for purity on the majority of datasets. Indeed, the obtained clusters have a better homogeneity than K-medoid and Dbscan and K-means. It is obvious in Figure. 4 and Figure. 5 that ClusSMOG is able to make good decisions in the assignment of objects, which leads to high values of F-measure and Rand Index obtained by our algorithm in all datasets. Thus, overall, ClusSMOG presents definitely better performance than others algorithms and similar to K-means, in addition our technique automatically calculates the number of the clusters while K-means requires the desired number of clusters as input.

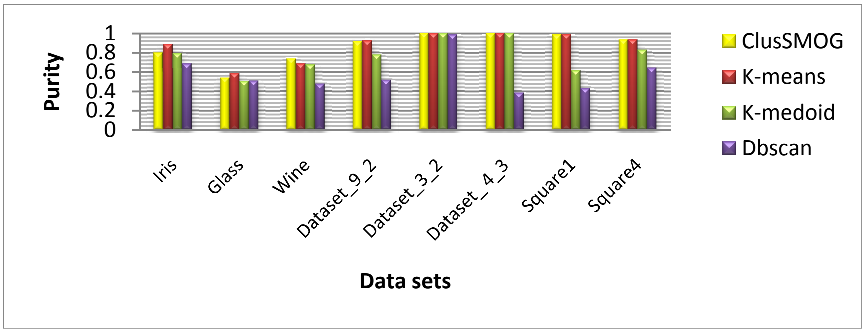


Fig. 3. Purity value comparison between different algorithms

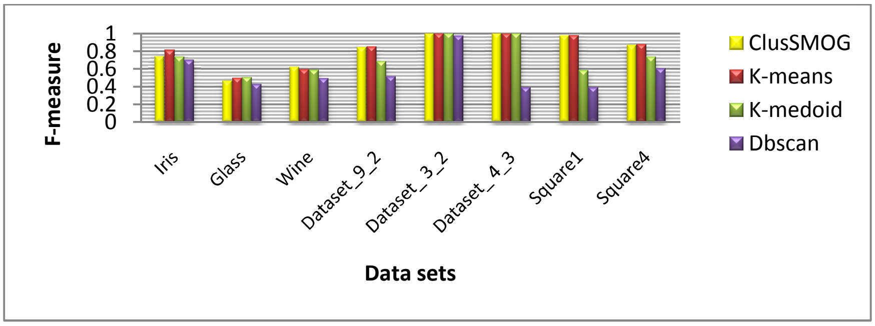


Fig. 4. F-measure value comparison between different algorithms

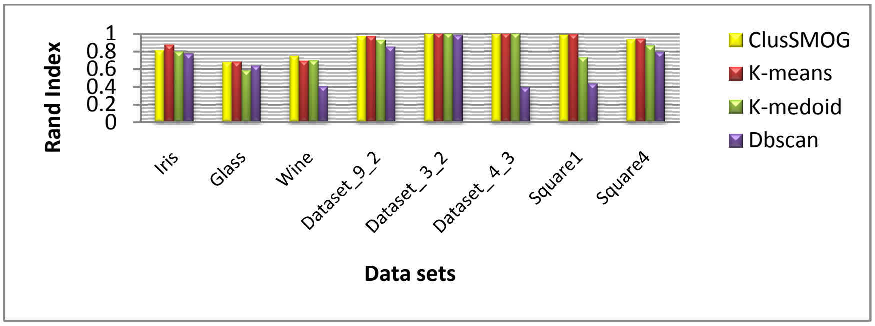


Fig. 5. Rand Index value comparison between different algorithms

In order to analysis the proposed clustering technique more accurately, we use silhouette metric as a method of interpretation and validation of clustered data as it is shown in Figure. 6. ClusSMOG clusters the data objects with high inter-cluster and low intra-cluster. One may have high silhouette value for K-means in some cases, this happens due to fact that the silhouette metric considers only intra-cluster inertia. Since K-means is a single objective clustering method, it optimises intra-cluster inertia effectively, especially when it is not easy to consider both objectives. However, in most cases, the presented algorithm gives higher silhouette metric, which indicates the effectiveness of the ClusSMOG.

From these results, ClusSMOG has proven its robustness compared to other mono/multi objectives algorithms using various evaluation measures. It gives solutions to clustering's problems by using the concepts of sequential games theory with a cluster initialisation mechanism, which plays a very important role on the final result of clustering.

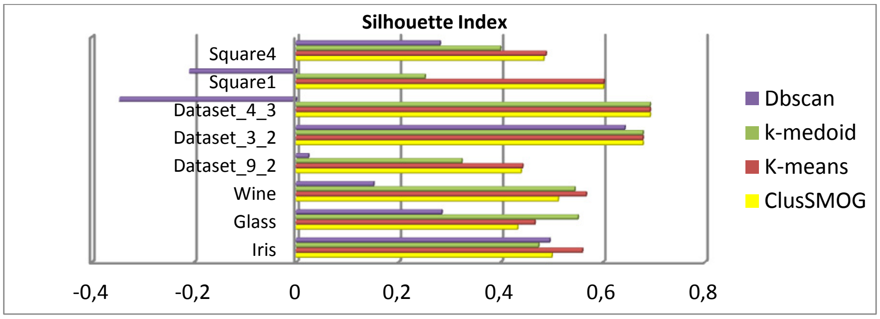


Fig. 6. Silhouette Index comparison between different algorithms

The proposed approach is also analysed on the basis of complexity metric. As sequential games are represented in a tree form, the complexity of a tree browsing is $O(m)$, where m is the number of nodes. In the second step where we have formulated multi-act sequential two-player games, the size of a player's actions is limited to $|X_i| = L * |C_i|$. Assuming that the average size of each cluster $|C_i| = \left\lfloor \frac{n}{K^{(0)}} \right\rfloor$ and $conflict^{(t)} = K^{(0)}$. Hence, in the worst-case, the complexity is $O(m * n * \sqrt{n})$, where $m \leq n^2$. Assuming that the third step is carried out also in the worst-case, with $K^{(0)}$ single-act sequential games, the complexity is $O(n * \sqrt{n})$.

4 Conclusion

We proposed a novel microeconomic-theory-based technique for simultaneous multi-objective clustering based on conflicting objectives, intra-cluster / inter-cluster inertia and connectivity with automatic K-determination. Our methodology is based on the backward induction in order to derive a desirable fairness in the final clustering results. The proposed technique is also able to determine the appropriate number of cluster dynamically. For this, we developed an interesting and very important cluster initialisation mechanism that has direct impact on the final clustering results. The experimental study conducted on some well-known benchmarks datasets provided important insights on the performance of the game theoretic algorithms. Experimental results confirm the effectiveness of the proposed approach over state-of-the-art clustering algorithms (including single and multi-objective techniques. As future work, we will look at some parts of the technique where we can use parallelism or concurrent actions.

References

1. Law, M.H.C., Topchy, A.P., Jain, A.K.: Multiobjective Data Clustering. In: Computer Society Conference on Computer Vision and Pattern Recognition (2004)
2. Strehl, A., Ghosh, J.: Cluster Ensembles: A Knowledge Reuse Framework for Combining Multiple Partitions. Machine Learning Research 3, 93–98 (2003)
3. Dempster, A., Laird, N., Rubin, D.: Maximum Liklihood From Incomplete Data via the EM Algorithm. Journal of The Royal Statistical Society, Series B 39, 1–38 (1977)
4. Handl, J., Knowles, J.: MultiObjective Clustering Around Medoids. In: Proceedings of IEEE Congress on Evolutionary Computation, pp. 632–639 (2005)
5. Handl, J., Knowles, J.: An Evolutionary Approach to Multiobjective Clustering. IEEE Transaction on Evolutionary Computation 11, 56–76 (2007)
6. Von Neumann, J.: Zur Theorie der Gesellschaftsspiele. Mathematische Annalen 100, 295–320 (1928)
7. Gupta, U., Ranganathan, N.: A game theoretic approach for simultaneous compaction and equipartitioning of spatial data sets. IEEE Transcation on Knowledge and Data Engineering 22, 465–478 (2010)
8. Badami, M., Hamzeh, A., Hashemi, S.: An enriched game-theoretic framework for multi-objective clustering. Applied Soft Computing 13, 1853–1868 (2013)
9. Garg, V.K., Narahari, Y., Murty, M.N.: Novel Biobjective Clustering (BiGC) based on Cooperative Game Theory. IEEE Transactions on Knowledge and Data Engineering (2013)
10. Bulo, S.R., Pelillo, M.: A game-theoretic approach to hypergraph clustering. Advances in Neural Information Processing Systems (2009)
11. Goutte, C., Toft, P., Rostrup, E., Nielsen, F.Å., Hansen, L.K.: On Clustering fMRI Time Series. NeuroImage 9, 298–310 (1999)
12. Handl, J., Knowles, J.D.: Evolutionary Multiobjective Clustering. In: Yao, X., et al. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 1081–1091. Springer, Heidelberg (2004)
13. Tufféry, S.: Data Mining et Statistique Décisionnelle: L'intelligence des données (2010)
14. `http://personalpages.manchester.ac.uk/mbs/Julia.Handl/mock.html`
15. Bandyopadhyay, S., Saha, S.: GAPS: A Clustering Method Using a New Point Symmetry-Based Distance Measure. Pattern Recognition 40, 3430–3451 (2007)
16. `http://www.isical.ac.in/~sanghami/data.html`

17. `http://archive.ics.uci.edu/ml/datasets.html`
18. RapidMiner website, `http://www.rapidminer.com`
19. Rand, W.: Objective Criteria for the Evaluation of Clustering Methods. J. Amer. Statist. Assoc. 66, 846–850 (1971)
20. Hubert, A.: Comparing partitions. J. Classification 2, 193–198 (1985)
21. van Rijsbergen, C.: Information Retrieval, 2nd edn. Butterworths (1979)
22. Rousseeuw, P.J.: Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis. J. Comput. Appl. Math. 20, 53–65 (1987)