Relational Data Partitioning using Evolutionary Game Theory

Lawrence O. Hall, Fellow, IEEE, Alireza Chakeri
Department of Computer Science and Engineering
University of South Florida
Tampa, Florida
Email: hall@cse.usf.edu, chakeri@mail.usf.edu

Abstract—This paper presents a new approach for relational data partitioning using the notion of dominant sets. A dominant set is a subset of data points satisfying the constraints of internal homogeneity and external in-homogeneity, i.e. a cluster. However, since any subset of a dominant set cannot be a dominant set itself, dominant sets tend to be compact sets. Hence, in this paper, we present a novel approach to enumerate well distributed clusters using a distance function. For instance, X can convert an numerical data dissimilarity matrix. Specifically, a numerical object data can be the Euclidean distance, where each x_{ij} provides a feature value. On the other hand, relational data consists n^2 similarities (dissimilarities) between pairs of objects in O. The relational data are usually represented by an n x n similarity matrix A = (a_{ij}) (or dissimilarity matrix D = (d_{ij})), such that a_{ij} (d_{ij}) is the similarity (dissimilarity) between object i and object j. One can convert an numerical data X into a dissimilarity matrix D using a distance function. For instance, d_{ij} can be the Euclidean distance between objects i and j, i.e. d_{ij} = ||x_i - x_j||. Hence, a relational clustering algorithm can be applied to both types of data. However, there are relational data sets that do not come from a numerical object data. Consequently, one can only use relational data clustering methods in these situations.

A classical approach to relational data clustering is to use the concepts and approaches from graph theory [1]. In this regard, the data points are mapped to the vertices of a weighted edge graph such that the edge weights represent the similarities between objects of the corresponding nodes. Hence, these approaches map clustering problems to graph theoretic ones in which powerful algorithms have been developed including minimum spanning tree [2], and minimum cut [3], [4]. Recently in [5], the notion of a dominant set was introduced as a definition of a cluster arising from the generalization of maximal cliques in weighted edge graphs. The underlying assumption is that a cluster can be seen as a group of objects having more similarity to one another than objects not in the cluster. The two most important features of this approach are that it provides a measure of a cluster cohesiveness as well as a degree of belonging of every vertex to each cluster.

A dominant set is a subset of data points satisfying the constraint of internal homogeneity and external in-homogeneity. In [5], a very significant relation between dominant sets and the standard quadratic optimization problem was established. In particular, it was proved that dominant sets are strict local solutions of the corresponding quadratic problem. In addition, it is well known from evolutionary game theory that strict local solutions are in one-to-one correspondence with evolutionary stable strategies (ESSs) [6] of the game that has the similarity matrix of the data points as its payoff. In this regard, replicator dynamics, a class of evolutionary game-theoretic algorithms inspired by Darwinian selection processes [6], [7] provide a very efficient tool to find dominant sets. However, since evolutionary dynamics will find one equilibrium depending on the initial state, an enumeration method of dominant sets is needed. A simple approach is to use a peeling-off strategy [5], where a dominant set is detected, and then all of its vertices is removed from the graph. Repeating this procedure in the remaining data, we obtain a partitioning of the data where each dominant set corresponds to a cluster. However, since the graph is changing during every step of this method, one can easily argue that the resulted dominant sets may not exist in the original graph. To overcome the problem, in [8] and [9], two methods were introduced to enumerate the equilibriums of any symmetric game. In [8], after extraction of ESS, the graph is changed is such a way that the located ESS becomes unstable under the dynamics without affecting the remaining ESSs. In [9], a tabu-set method was developed using InImDynamics.
[10]–[12] in order to heuristically search the solution space.

On the other hand, dominant sets tend to be a compact set of vertices. This is because any subset of vertices of a dominant set cannot be a dominant set itself. In other words, based on the evolutionary game theory terminology, any subset of non zero elements of an ESS can not be even a Nash equilibrium (and consequently be an ESS). This property may cause a (big) cluster to be subdivided into several dominant sets. Hence, we propose an approach to efficiently search the solution space in order to find the desired number of well distributed dominant sets. In this regard, the vector containing the payoffs of every vertex playing against the equilibrium is partitioned into two subsets and the partition that does not contain the equilibrium vertices is used for detecting the other dominant sets. Also, when the number of clusters is not provided, we propose an efficient approach to extract the appropriate number of clusters using game theory relations. Then the same efficient approach to extract the appropriate number of clusters when the number of clusters is not provided, we propose an efficient approach to extract the appropriate number of clusters using game theory relations. Then the same method is applied to each of the partitions obtained. Although the clusters in each partition are clusters of their equivalent sub graph, they may not be clusters of the entire graph. Hence they are checked for being a cluster of the entire graph and if they are not, they are extended to be a cluster of the entire graph using InlmDyn.

II. Preliminaries

This section contains some background on dominant sets and evolutionary game theory, which is needed to develop the presented approach.

A. Dominant Sets

A classical approach to relational data clustering is to map the objects to the vertices of a weighted edge graph such that edge weights are the similarities between objects of the corresponding nodes. Let’s consider an undirected weighted edge graph $G = (V, A)$, where $V = \{v_1, \ldots, v_n\}$ is the set of vertices and $A = (a_{ij})$ is the similarity matrix in which $a_{ij}$ is the edge weight between vertices $v_i$ and $v_j$. Recently in [5], the notion of a dominant set was introduced as a definition of a cluster. In [5], it was proved that in an unweighted edge graph, dominant sets are equal to strictly maximal cliques. In this subsection, the fundamentals of the dominant set framework are presented briefly.

Let $S \subseteq V$ be a subset of vertices. The average weighted degree of $v_i \in V$ with regard to $S$ is defined as

$$awdeg_S(v_i) = \frac{1}{|S|} \sum_{v_j \in S} a_{ij}$$  \hspace{1cm} (1)$$

Also, if $v_j \notin S$, then a measure which reflects the relative similarity between nodes $v_j$ and $v_i$ with regard to the average similarity between node $v_i$ and its neighbors in $S$ is defined as

$$\phi_S(v_i, v_j) = a_{ij} - awdeg_S(v_i)$$  \hspace{1cm} (2)$$

As a result, the weight of $v_i \in S$ with regard to $S$ is defined as

$$w_S(v_i) = \begin{cases} 1 & \text{if } |S| = 1 \\ \sum_{v_j \in S \setminus \{v_i\}} \phi_S(v_i, v_j) \cdot w_{S \setminus \{v_i\}}(v_j) & \text{otherwise} \end{cases}$$  \hspace{1cm} (3)$$

while the total weight of $S$ is defined as follows

$$W(S) = \sum_{v_i \in S} w_S(v_i)$$  \hspace{1cm} (4)$$

Consequently, in [5], the following definition of the dominant set representing the notion of a cluster in a weighted edge graph was presented.

**Definition 1**: A non empty subset of vertices $S \subseteq V$ such that $W(T) > 0$ for any non-empty subset $T \subseteq S$, is a dominant set if it has the two following conditions:

1) $w_S(v_i) > 0$, for all $v_i \in S$.
2) $w_{S \cup \{v_i\}}(v_i) < 0$, for all $v_i \notin S$.

It can be seen that conditions 1 and 2 correspond to the two main characteristics of a cluster, i.e. internal homogeneity and external in-homogeneity, respectively.

Now consider the following quadratic program

$$\begin{align*}
\max_{x} & \quad f(x) = x^T A x \\
\text{subject to} & \quad x \in \Delta
\end{align*}$$  \hspace{1cm} (5)$$

where $\Delta = \{x \in \mathbb{R}^n : x \geq 0, \sum x_i = 1\}$ is the standard simplex, and $A$ is an $n \times n$ symmetric similarity matrix of graph $G$. Pavan et al [5] established a one to one correspondence between dominant sets and strict local maximizers of a quadratic form over the standard simplex. Particularly, if $S \subseteq V$ is a dominant set, then its characteristic vector $x^S$, which is defined as

$$x^S_i = \begin{cases} w_S(v_i) & \text{if } v_i \in S \\ 0 & \text{otherwise} \end{cases}$$  \hspace{1cm} (6)$$

is a strict local solution of program (5) with $A$ as being similarity matrix of the weighted edge graph $G$. Conversely, if $x$ is a strict local solution of program (5), then its support $S = \sigma(x)$, where $\sigma(x) = \{i \in \{1, \ldots, n\} : x_i > 0\}$, will be a dominant set.

B. Game Theoretic Viewpoint of Dominant Sets

A novel connection between finding dominant sets using quadratic programming and evolutionary game theory was established in [5], [13]–[16]. In this subsection, the basic definition of evolutionary game theory is briefly reviewed.

A strategic game consists of a set of players, their strategies, and payoffs available for all combinations of players’ strategies. In evolutionary game theory, the players are repeatedly drawn from a random infinite population to play the game. Let us consider a large population of individuals who are randomly matched to play a symmetric two-player game (doubly symmetric game) $G = (S, A)$. Each player plays a certain strategy from strategy set $S = \{1, 2, \ldots, n\}$ with $A = (a_{ij})$ the $n \times n$ symmetric payoff matrix, which assigns
a payoff \( a_{ij} \) to a player playing the strategy \( i \) against a player playing strategy \( j \).

The state of the population can be represented by an \( n \)-dimensional vector \( x = (x_1, x_2, \ldots, x_n) \), where \( x_i \) corresponds to the fraction of individuals in the population playing strategy \( i \in S \), such that \( \sum_{i=1}^{n} x_i = 1 \). This set is formally identical to the set of mixed strategies, and consequently it belongs to \( \Delta \). Hence, the expected payoff received for a player using mixed strategy \( y \) against an opponent playing mixed strategy \( x \) is \( u(y, x) = y^T Ax \).

The best response for a player is a choice that maximizes the payoff of that player given the actions taken by the other players. Consequently, the best responses against a mixed strategy \( x \) are the set of mixed strategies \( \beta(x) = \{ y \in \Delta \mid u(y, x) = \max_{z \in \Delta} u(z, x) \} \). A mixed strategy \( x \) is a symmetric Nash equilibrium if and only if it is the best response to itself, i.e. \( u(y, x) \leq u(x, x) \), \( \forall y \in \Delta \). This implies that for all \( i \in \sigma(x) \), where \( \sigma(x) = \{ i \in S \mid x_i > 0 \} \) is called the support of mixed strategy \( x \). \( u(e^i, x) = u(x, x) \). In other words, the payoff for every strategy in the support of \( x \) is the same, while all strategies outside the support of \( x \) earn a payoff less than or equal to \( u(x, x) \) which is called the equilibrium payoff. It is easy to see that all KKT points of the problem (5) with matrix \( A \) are the same as the Nash equilibriums of the corresponding game, and vice versa.

Now assume that the whole population plays mixed strategy \( x \) and a small group of mutants enter this population and adopt a different mixed strategy \( y \). A mixed strategy \( x \) is called an Evolutionary Stable Strategy (ESS) if and only if:

1) It is a Nash equilibrium, i.e. \( u(y, x) \leq u(x, x) \), \( \forall y \in \Delta \) (equilibrium condition).

2) If \( u(y, x) = u(x, x) \), then \( u(y, y) \leq u(x, y) \) (stability condition).

Now, in [13], a relation between ESS of two player games with payoff matrix \( A \) and the quadratic program (5) was established, such that \( x \) is the ESS of a two player game with payoff matrix \( A \) if and only if it is a strict local solution of problem (5). As a result, based on section II.A, one can find the ESS of a game with payoff matrix \( A \) to find the dominant sets of a graph with similarity matrix \( A \).

On the other hand, replicator dynamics, a class of evolutionary game theory dynamics, provide us a very efficient tool to find the strict local solutions. It was proved that in [6], the asymptotically stable points of the replicator dynamics are in one-to-one correspondence of the strict local maximizer of problem (5). In other words, \( x \in \Delta \) is asymptotically stable under replicator dynamics if and only if \( x \in \Delta \) is a strict local maximizer of problem (5), and if and only if \( x \in \Delta \) is an ESS for the equivalent symmetric game [6]. Specifically, it was proved that replicator dynamics converge to a Nash equilibrium if the initial population state starts from the interior of \( \Delta \) represented by \( \text{int}(\Delta) = \{ x \in \Delta \mid x_i > 0, \forall i \} \).

However, in this paper, we use a new class of dynamic called InImDyn (Infection and Immunization Dynamics) which can overcome some limitations of the replicator dynamics [11], [12]. Every fixed point of this dynamic is a Nash equilibrium and vice versa. Specifically, any trajectory starting in \( \Delta \) converges to a Nash equilibrium; in other words, the limits points of this dynamic are Nash equilibriums (also, there is a one to one correspondence between ESSs and the asymptotically stable points of InImDyn). In addition, from a computational viewpoint, every iteration in InImDyn is linear in the number of strategies enumerated compared to the quadratic one in replicator dynamics [10].

### III. Efficient Dominant Set Partitioning for Relational Data

To use the dominant set clustering method when numerical object data \( X \) is provided, one converts the obtained dissimilarity matrix \( D \) to a similarity matrix \( A \). As a result, it is common to use the following transformation:

\[
    a_{ij} = \exp(-d_{ij}/\sigma^2)
\]

where \( \sigma \) is a positive real number which effects the rate of decrease rate of the similarity values. Although we expect a clustering method to be invariant to the changes in similarity matrices caused by different \( \sigma \)'s and yield similar clustering results, but the dominant set clustering is not robust. The results of the clustering vary widely with different \( \sigma \)'s.

In clustering problems, two possible cases happen regarding the number of clusters in a data set. The first is when the number of clusters is known in advance (we call it "known number of clusters") which is more common in numerical data than relational data. The second situation is when we need to determine the appropriate number of clusters (we call it "unknown number of clusters situation"). In this paper, we tackle both situations using evolutionary game theory relations.

Now, to lay the theoretical groundwork for our algorithm, we further analyze the properties of the Nash equilibriums and InImDyn. Let’s denote \( G_B \) as the subgraph of \( G \) induced by the set of vertices in \( B \subset S \) and \( \hat{x} \) as the expansion of mixed strategy \( x \) of a subgraph \( G_B \) to \( G \) by adding zero entries to \( x \) for vertices that do not belong in \( G_B \). If \( x \) is a Nash equilibrium of the subgraph \( G_B \), it is easy to judge whether \( \hat{x} \) is a Nash equilibrium of \( G \) [17]. In this regard, if all other vertices outside \( G_B \) playing against \( \hat{x} \) earn payoffs not larger than equilibrium payoff \( u(x, x) \), then \( \hat{x} \) is a Nash equilibrium of the entire graph \( G \). Hence, if there is a vertex \( i \) such that it earns a higher payoff playing against \( \hat{x} \) than equilibrium payoff \( u(x, x) \), \( \hat{x} \) is no longer a Nash equilibrium of the whole graph \( G \). We use the above analysis to develop our proposed algorithm. Specifically, after finding each equilibrium of a subgraph, the algorithm checks whether the obtained equilibrium is an equilibrium of the whole graph. If it is not, the obtained equilibria of the subgraph is used as the initial point for the InImDyn to run on the whole graph. Now, the important feature of InImDyn, as stated in subsection II.B, that makes it appropriate for our approach is that there is a one to one correspondence between all of its fixed points and Nash equilibriums. In other words, in spite of replicator dynamics where the trajectory needs to be started in the interior of the simplex to converge to a Nash equilibrium, the InImDyn provides convergence to a Nash equilibria starting from a point in the simplex.

#### A. Known number of clusters

For the situation in which the number of clusters \( c \) is provided, using the notion of dominant sets is limited to a
data set that has a number of equilibriums greater than or equal the provided number of clusters. However, this limitation is generally satisfied for most data sets (This is because of the fact that recently in [18], it was proved that in random two player games in which each player has the same number of strategies \(|S|\), the mean number of Nash equilibriums is \( \exp(|S| (B + O(\log|S|))) \) where \( B \approx 0.286144 \)). Hence, a naive approach is to enumerate dominant sets and consider the first \( c \) dominant sets created. Particularly, after finding each dominant set, the peeling-off strategy is employed with the difference that if the obtained equilibriums of the subgraph is not the equilibrium of the whole graph, it is used as the initial point for the InImDyn to run on the whole graph [9]. But, as we show in the experimental section, the obtained dominant sets might not be well distributed and they may belong to the same clusters. Hence, we propose an approach to extract the informative dominant sets and then efficiently assign the remaining objects to them. Here, the informative dominant sets mean highly distributed ones, i.e. their supports are mutually disjoint sets if possible.

In order to find desired number of well distributed equilibriums, we developed a method to search the solution space appropriately. In this regard, after finding each equilibrium, the vector containing payoffs for each vertex playing against the equilibrium strategy is computed and then it is discretized in order to remove the vertices close to the equilibrium vertices. In fact, we are doing this in the hope of removing the vertices in the basin of attraction of the obtained equilibrium. One can use several discretization criteria that are used in spectral graph methods for image segmentation [4], [19] such as median cut, jump cut and ratio cut [20]. In this paper, we use the jump cut method. For jump cut, we sort the payoff vector entries in decending order and find the largest difference between two entries in the sequence. The entries before these two entries, which have high payoffs, are grouped as close vertices to the vertices in the equilibrium. It is easy to see that vertices in the equilibrium should be included in this group since they have the highest payoff value. By doing this, close vertices to the ones in the cluster as well as the vertices in the cluster will not be considered to find the next equilibrium.

The proposed approach is summarized in Algorithm 1. It takes a graph \( G = (S, A) \) (\( S \) is the set of vertices and \( A \) is the similarity matrix) as input and returns a set \( N \) containing desired number \( c \) clusters of \( G \) and two sets of vertices that are explained below. The enumeration algorithm uses InImDyn (function InImDyn in line 6) that is initialized to the slightly perturbed barycenter of \( \Delta_{S,T} \) (function Barycenter in line 5). Also, a tabu-set \( T \) is used to explore the solution space more efficiently. After finding each equilibrium, its support and those comparable vertices obtained by the jump cut method are added to \( T \) (function JumpCut in line 11), and then InImDyn run on the remaining objects, i.e on the subgame obtained by strategies \( S \setminus T \). If the resulting equilibrium is not an equilibrium of the entire game, it is used as the initial point for InImDyn to run again on the whole graph \( G \). This procedure repeats until \( c \) clusters are enumerated or \( T \) becomes equal to \( S \). In the latter case, one can use a more strict condition for constructing \( T \) such as selecting the vertices whose payoffs are bigger than the average plus the variance of all vertices playing against the equilibrium. The algorithm also gives us two more sets. Let’s assume that there are \( c \) clusters represented by the set of equilibriums \( N = \{n_1, n_2, ..., n_c\} \). This set partitions the vertices into two subsets \( Z = \{j \in S | j \in \sigma(n_i), \exists i\} \) and \( \tilde{Z} = \{j \in S | j \notin \sigma(n_i), \forall i\} \) that the set of objects that do not belong to any equilibrium, respectively.

**Algorithm 1. Equilibrium Enumeration**

1) **Input**: Graph \( G = (S,A) \), and \( c \) as the number of clusters.
2) **Output**: \( N \) a set containing at most \( c \) Nash equilibriums of \( G \), \( Z \) and \( \tilde{Z} \).
3) Initialize \( N = \emptyset, T = \emptyset \) and \( \text{counter} = 1 \).
4) while \( \text{counter} \leq c \) or \( T \neq S \)
   5) \( b = \text{Barycenter}(G_{S,T}) \)
   6) \( x = \text{InImDyn}(G_{S,T}, b) \)
   7) if \( \text{there exists} \ i \in S \ \text{such that} \ u(e_i, \hat{x}) > u(\hat{x}, \hat{x}) \)
   8) \( y = \text{InImDyn}(G_{S}, \hat{x}) \)
   9) else
   10) \( y = \hat{x} \)
   11) \( T = T \cup \text{JumpCut}(y) \)
12) Add \( y \) to \( N \), and \( \text{counter} = \text{counter} + 1 \)
13) Partition the vertices into \( Z = \{j \in S | j \notin \sigma(n_i), \forall i\} \) and \( \tilde{Z} = \{j \in S | j \in \sigma(n_i), \exists i\} \) induced by \( N \)

Based on Algorithm 1, every mixed strategy Nash equilibrium \( x \) provides a cluster in which its support represents the objects belong to that cluster. In addition, the corresponding equilibrium payoff \( u(x, x) \) shows the cohesiveness (compactness) of the cluster. Now, consider the mapping \( C : S \to \{1, ..., c\} \) that assigns each object a class label. To partition the data, we propose the following relation to assign object \( i \) to the appropriate partition

\[
C(i) = \begin{cases} 
\text{argmax}_j u(e_i, n_j) & \text{if } i \in \tilde{Z} \\
\text{argmax}_j n_j(i) & \text{if } i \in Z 
\end{cases} \tag{8}
\]

where \( C(i) \) represents the partition assigned to object \( i \). The intuition behind this relation is that every object \( i \) in \( \tilde{Z} \) is assigned to the partition in which it gains higher relative similarity with regard to the cluster cohesiveness. In other words, although it is necessary to gain high payoff playing against the equilibrium strategy, but it also depends on the equilibrium payoff. Also, every object \( i \) in \( Z \) belongs to the partition that has highest degree of participation. The partitioning approach is summarized in Algorithm 2.

**Algorithm 2. Dominant Sets Partitioning**

1) **Input**: Graph \( G = (S,A) \), the set of clusters \( N = \{n_1, ..., n_c\} \), \( Z \) and \( \tilde{Z} \).
2) Initialize the vector \( C \) of size \( n \).
3) for each \( i \in \{1, ..., n\} \)
   4) if \( i \in Z \)
   5) \( C(i) = \text{argmax}_j n_j(i) \)
   6) else if \( i \in Z \)
   7) \( C(i) = \text{argmax}_j u(e_i, n_j) \)
8) **Output**: The vector \( C \) representing the partition number of each object

**B. Unknown number of clusters**

There are several approaches to determine the appropriate number of clusters in a data set [21]. In this paper, Dunn’s
index is used to identify sets of clusters that are compact, with a small variance between members of the clusters, and well separated as compared to the within cluster variance. We develop a hierarchical clustering method that employs algorithms similar to Algorithm 1 and 2 to enumerate the equilibriums of the graph using Dunn’s index.

First, Dunn’s index is used to determine the appropriate number of clusters in the first level of the hierarchy using Algorithm 1, and data is partitioned using Algorithm 2. Then, each partition is considered as an independent data set and the same routine is applied, with the difference that clusters in each partition are extended to be clusters of $G$ (if they are not already a clusters of $G$). This hierarchical equilibrium enumeration continues until the desired level of hierarchies is reached.

The following summerizes the proposed approach in three steps. To simplify, we assume that the minimum ($\text{min}$) and maximum ($\text{max}$) number of clusters to use Dunn’s index is the same for all levels. The $\text{min}$ and $\text{max}$ values are defined apriori by the user.

1) Apply the Algorithm 1 for all number of clusters between $\text{min}$ and $\text{max}$.
2) Find the best number of cluster using Dunn’s index.
3) Apply Algorithm 2 and go to step 4 if the number of levels is not reached, otherwise terminate.
4) Do step 1 for each partition in step 3.

The first evidence which is congruent with our experimen-
Fig. 3: (a) Three dominant sets, represented by ‘×’, ‘o’ and ‘+’, were extracted based on Algorithm 1; (b) Partitioning data points to the obtained dominant sets in (a) using Algorithm 2; (c) Partitioning the data points to the two equilibriums in the level 1 hierarchy using ‘×’, ‘o’ and ‘+’; (d) Partitioning the data points to the four equilibriums in the level 2 hierarchy using ‘×’, ‘o’, ‘∗’ and ‘+’.

tal results is that the clusters obtained in a level generally will be obtained in the following levels. This is because of the fact that equilibriums obtained by InlmDyn are asymptotically stable points. Also, the clusters in each partition are usually obtained in decreasing order of size which can be an advantage of our method. In addition, for the first levels, equilibriums obtained in each subgraph are usually the equilibriums of the entire graph, hence there is no need to extend them to an equilibrium of the whole graph.

IV. EXPERIMENTAL RESULTS

In order to test the effectiveness of the proposed algorithm, we present a number of examples that illustrate its various facets in both numerical and relational data sets. All the algorithms were written in m code for Matlab and run on a machine equipped with 2 Intel 2 GHz CPUs and 8 GB RAM. For numerical data sets, we set $\sigma$ in (7) equal to the mean of the similarity values.

A. Three Gaussian Clouds

This example contains 600 object vectors $X \subset \mathbb{R}^2$. The data were drawn from three Gaussian distributions containing 450, 100 and 50 examples. Figure 1(a) depicts the similarity matrix obtained by (7) as an image. Each pixel of the image displays scaled similarity value of two objects between $[0, 255]$. White pixels represent high similarity, where black pixels represent low similarity. The cluster tendency is shown by the number of white blocks along the diagonal of the image. Hence, the image shows that there are three separable clusters represented by high similarities around the main diagonal. Figure 1(b) shows the situation where the first three equilibriums are considered as the desired three clusters. It can be easily seen that they are not distributed and they all belong to the same cluster. However, the result of the proposed approach in Algorithm 1 is shown in Figure 1(c). In this case, the obtained three equilibriums are well dispersed among the whole data, and each of them represents a true visualized cluster. As one can see there are some data points which do not belong to any clusters. Hence, Figure 1(d) shows the result of applying the Algorithm 2 to completely partition the data.

The vectors containing the payoffs of every vertex playing against the three equilibriums are sorted and plotted in Figure 2(a-c). In all three figures, vertices in the equilibriums have the maximum payoffs, and as a cluster gets smaller the corresponding maximum value gets smaller. Moreover, there exists a significant difference in payoffs of some of the vertices comparing to the maximum payoff, that makes the jump cut an appropriate method for discretization purposes.

B. Iris DataSet

The Iris plant data set consists of 50 samples from each of three species of Iris, i.e., Iris setosa, Iris virginica and Iris versicolor. Four numeric attributes (the length and the width of the sepals and petals) were measured from each plant. Visualization of the Iris data set indicates that one class is completely separable and the other two classes intersect to a degree. Figure 3(a) shows the results of enumerating three equilibriums using Algorithm 1. Also, Figure 3(b) depicts
assigning clusters to the remaining data points. It is noted that
the separable class (shown with symbol “+”) is well separated
on the lower-left corner and the two overlapped classes (“o”
and “×” symbols) are also separated effectively in the upper-
right corner.

Moreover, we also applied the unknown number of clusters
approach with 2 levels of hierarchies on this data set. For the
first level, the Dunn’s index suggests two equilibriums. The
result is shown in Figure 3(c) representing the two obtained
partitions. In the second level, Dunn’s index suggests two
equilibriums in each partition, where all of them are eventually
the equilibriums of the whole data set and hence there is
no need to extend them. Figure 3(d) depicts the new four
partitions. We see here that setosa is split into two clusters
which does not provide us information as the debate with this
data is whether there are two or three classes given the features.

C. Bioinformatics Data

The last example is the real world GPD194 [22], [23]
relational data set. This data set is not numerical. Rather,
dissimilarity relation values were obtained by a fuzzy measure
applied to annotations of 194 human gene products which
appear in the Gene Ontology. The data consists of 21, 87
and 86 gene products from Myotubularin, Receptor Precursor
and Collagen Alpha Chain protein families, respectively. The
dissimilarity matrix $D$ is converted to the similarity matrix $A$
as follows:

$$a_{ij} = 1 - \frac{d_{ij}}{\max(D)} \quad (9)$$

The three protein families are visible in Figure 4(a); they
have been sorted so that the upper left block is Myotubularin,
the middle block is Receptor Precursor, and the lower right
block is Collagen Alpha Chain. The hierarchical partitioning
approach is applied on this data set. The highest Dunn’s index
value in level 1 gives 2 clusters which are shown in Figure 4(b).
The partitions induced by the obtained two clusters are shown
in Figure 4(c) where the data points are sorted according to
the cluster that they belong to. The results of the second level
of hierarchy are shown in Figures 4(d) and (e). Using Dunn’s
index, each partition contains two clusters which are shown in
Figure 4(d). The clusters were obtained in the decreasing order
of size in each partition. In Figure 4(e), for each partition, data
points are sorted by the decreasing order of size of the clusters
to which they belong. We can see that the obtained clusters are
well distributed and represent the whole structure of the data.
Also, the clusters in first level are the clusters of the second
level as we discussed before.
V. CONCLUSIONS

It is well known that, since dominant sets tend to be compact, there can be many dominant sets in a graph. Also, there can be some data points that are not in the support of any dominant sets. Also, the definition of a dominant set is not robust one and, consequently, the similarity matrix relies heavily on the parameter \( \sigma \). To overcome these problems, in this paper an equilibrium enumeration algorithm was introduced using InImDyn to extract well distributed dominant sets with the number of clusters either known or unknown. Furthermore, based on game theory relations, a simple and efficient method was proposed to put unassigned objects into the appropriate clusters.

REFERENCES


