A New Gravitational Clustering Algorithm

Jonatan Gomez[*] Dipankar Dasgupta[†] Olfa Nasraoui[‡]

Abstract

This paper presents a novel unsupervised robust clustering technique based on the Gravitational Law and the second Newton's motion Law. The technique automatically determines the number of clusters in the target data set. Basically, each data record in the source data set is considered as an object in the feature space. The objects are moved by using the gravitational force and the second motion law. Because the clusters define a disjoint collection of sets, an optimal disjoint set union-find structure is used to store and update the clusters that are being conformed by closest objects. The proposed technique is robust to noise, can be used to generate a partition of the data set at multiple resolution levels, and can also be used to extract seeds to form a good summary of the data. Experiments with synthetic and real data were conducted to show the performance of the proposed clustering technique.

keywords: Clustering, Gravitational, Robust, Unsupervised, Scalable

1 Introduction

Clustering is an unsupervised learning technique that takes unlabeled data points (data records) and classifies them in different groups or clusters. This is done in such a way that points assigned to the same cluster have high similarity, while the similarity between points assigned to different clusters is low [12]. Many clustering techniques have relied on the assumption that a data set follows a certain distribution and is free of noise. In fact, if noise is introduced, several techniques based on the least square estimate are spoiled [19]; such is the case of k-means [16] and fuzzy k-means [3]. Several approaches have tried to tackle this problem; some of them based on robust statistics [11, 19], and others based on modifying the objective of the fuzzy centroid mean to make the parameter estimate more robust to noise [14, 2, 8]. In all of these methods, the number of cluster is given in advance. When the number of clusters is not known, the clustering techniques are called unsupervised [13, 9, 18].

We propose a novel unsupervised robust clustering technique based on the Gravitational Law and the second Newton's motion Law. Basically, each data record in the source data set is considered as an object in the feature space and are moved by using the gravitational force and the second motion law. Gravitational concepts have been applied to visualize clustering results [5] and for clustering analysis [22]. The gravitational clustering algorithm proposed by Wright in [22] is an hierarchical agglomerative algorithm: the gravitational forces are used as mechanisms for merging particles (data points) until only one particle (cluster) remains in the system. The following is a list of the most important properties of the dynamic simulation proposed by Wright:

- To determine the new position of a single particle all the remaining particles in the data set are used.
- When two particles (initially data points) are enough close to be merged, one of them is removed from the simulation and the mass of the other is increased with the mass of the particle removed
- There is a maximum distance that each data point can move in each iteration of the algorithm. This is a parameter that has to be fixed by the user.
- As mentioned previously, the simulation is terminated when only one particle remain in the system.

In this paper we propose a new dynamic simulation strategy (based on gravitational clustering) that can be used as a clustering algorithm. The main advantages over the Wright techniques are speed, robustness and unsupervised (it does not need to know the number of clusters in advance). This paper is divided in four sections. Section 1 presents the basic theoretical aspect of Newton motion laws, the gravitational law and the optimal disjoint union-find structure used in the development of the proposed approach. Section 2 describes the proposed approach, compares it with agglomerative hierarchical clustering techniques, and gives its computational complexity. Section 3 presents the experiments performed over synthetic and real data sets, along with their results and analysis. Section 4 draws some conclusions.

1.1 Newton Motion Laws Kinematics is the science of describing and explaining the motion of objects

[*]The University of Memphis - Department of Mathematical Sciences, Computer Sciences Division and Universidad Nacional de Colombia

[†]The University of Memphis - Department of Mathematical Sciences, Computer Sciences Division

[‡]The University of Memphis - Department of Electrical & Computer Engineering

in an universe by using some models ([6, 21, 17, 20]). Isaac Newton (1642-1727) developed a motion model, **The Newton Motion Laws**, that describe and explain with high precision the motion of macro-objects in our Universe, but cannot explain and describe the motion of micro-objects. This section will present the Newton motion laws for a moving object describing a straight line trajectory in the n-dimensional Euclidean space.

1.1.1 1-Dimensional Motion Laws Let be x an object in the Euclidean space, and t be a real number representing an instant in time. Let $x(t)$ be the object position at time t, $s(t)$ is the object speed at time t, $a(t)$ is the object acceleration at time t, and $F(t)$ is the force applied on the object at time t. The speed and acceleration of an object are defined respectively as:

$$s(t) = \frac{dx(t)}{dt} \tag{1.1}$$

$$a(t) = \frac{ds(t)}{dt} = \frac{d^2x(t)}{dt^2} \tag{1.2}$$

The speed of an object is the derivative function of the position function, and the acceleration is the derivative function of the speed function. If the speed or acceleration functions of an object is known, the position can be deduced as:[1]:

$$s(t) = \int_0^t a(t)dt \tag{1.3}$$

$$x(t) = \int_0^t s(t)dt = \int_0^t \left[\int_0^t a(t)dt\right] dt \tag{1.4}$$

If the acceleration of the object is a constant function then, the speed and position are:

$$s(t) = s(0) + at \tag{1.5}$$

$$x(t) = x(0) + s(0)t + \frac{at^2}{2} \tag{1.6}$$

where, $x(0)$ is the initial object position, $s(0)$ is the initial object speed, and a is the constant value for the object acceleration.

Equations (1.5) and (1.6) can be used to simulate the movement of an object when the position and speed functions cannot be determined analytically. In this way, the acceleration is considered constant during an interval of time $\triangle(t)$, and the speed and position of the object at time $t + \triangle(t)$ are approximated as:

$$s(t + \triangle(t)) = s(t) + a(t)\triangle(t) \tag{1.7}$$

$$x(t + \triangle(t)) = x(t) + s(t)\triangle(t) + \frac{a(t)\triangle(t)^2}{2} \tag{1.8}$$

Finally, if m_x is the mass of the object x, then the force exerted on the object is defined according to Newton's second motion Law as follows:

$$F(t) = m_x a(t) \tag{1.9}$$

1.1.2 n-Dimensional Motion Laws for straight line trajectories The n-dimensional motion laws for straight line trajectories are the vectorial extension of the 1-dimensional motion laws. Let x be an object in the n-dimensional euclidean space, that is moving in the direction given by the vector $\overrightarrow{d}$, and t be a real number representing an instant of time. Let $x(t)$ be the object position at time t, $v(t)$ be the object velocity at time t, and $\overrightarrow{a(t)}$ be the object acceleration at time t. The velocity and the acceleration vector are defined as the vectorial extension of (1.1) and (1.2) respectively:

$$v(t) = \frac{s(t)}{\left\|\overrightarrow{d}\right\|}\overrightarrow{d} \tag{1.10}$$

$$\overrightarrow{a(t)} = \frac{a(t)}{\left\|\overrightarrow{d}\right\|}\overrightarrow{d} \tag{1.11}$$

Where, $\left\|\overrightarrow{d}\right\|$ is the magnitude of the direction vector $\overrightarrow{d}$, and $a(t)$ and $s(t)$ are the acceleration and speed of the object in the moving direction $\overrightarrow{d}$. If the acceleration is constant, then the position and velocity are given by the vectorial extension of (1.5) and (1.6) respectively:

$$v(t) = v(0) + \overrightarrow{d}t \tag{1.12}$$

$$x(t) = x(0) + v(0) * t + \frac{\overrightarrow{d}t^2}{2} \tag{1.13}$$

If the acceleration is not constant, then the movement of an object can be approximated by using the vectorial extensions of (1.7) and (1.8) respectively:

$$v(t + \triangle(t)) = v(t) + \overrightarrow{a(t)}\triangle(t) \tag{1.14}$$

$$x(t + \triangle(t)) = x(t) + v(t)\triangle(t) + \frac{\overrightarrow{a(t)}\triangle(t)^2}{2} \tag{1.15}$$

[1]In general the acceleration function can be defined as a differential equation. Therefore, the speed and position functions are found by solving such a differential equation.

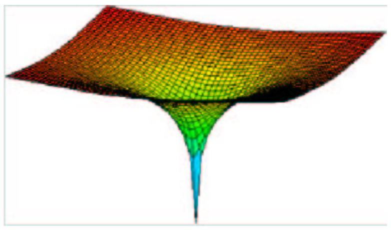
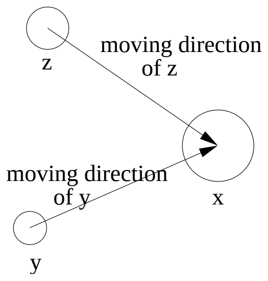
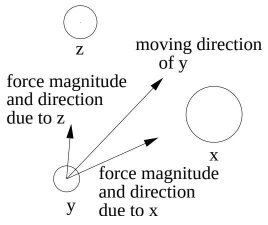


Figure 1: Gravitational field



(a) Movement directions of objects exposed to a gravitational field



z
moving direction of y
force magnitude and direction due to z
x
force magnitude and direction due to x
y

(b) Movement direction of an object under the gravitational field of two objects

Figure 2: Movement directions of objects exposed to gravitational fields

1.2 Gravitational Law These days, the most accepted theory for explaining the structure of the Universe and the motion of celestial bodies is the gravitational law developed by Isaac Newton. The Universal Law of Gravitation states that:

"Each body exerts an attractive force on any other body. The force is directed along the direction joining the centers of the two bodies. Its intensity is directly proportional to the product of their masses, and inversely proportional to the square of their center's distance."

The force exerted from one object x over another object y is expressed by the following equation:

$$F(t) = \frac{Gm_x m_y}{d(x(t), y(t))^2} \quad (1.16)$$

where, m_x and m_y are the masses of the two objects, $d(x(t), y(t))$ is the Euclidean distance between the two objects, and G is the Universal gravitational constant $(6.67 * 10^{-11})$.

The gravitational force exerted by an object over other objects defines a gravitational field around the object, as shown in Figure 1.

In this way, an object y is exposed to the gravitational field of object x, that pulls y toward the center of this field, with an intensity given by the gravitational force, and with the direction defined by the vector $\overrightarrow{d(t)} = x(t) - y(t)$. Using the direction vector $\overrightarrow{d}$, the gravitational force equation can be rewritten as:

$$F(t) = \frac{Gm_x m_y}{\left\|\overrightarrow{d(t)}\right\|^2} \quad (1.17)$$

Figure 2.a shows the movement directions of two objects (y and z) that are exposed to the gravitational field of a third object (x). In Figure 2.a, only the gravitational field of x is taking into account, i.e., the gravitational fields of y and z are not considered.

To calculate the final direction of an object's motion, it is necessary to calculate the magnitude of the gravitational force that is exerted on the object for every other object in the universe. Figure 2.b shows the final direction of motion and force magnitude of y according to the magnitude of the gravitational force exerted by x and z. The size of an object indicates its mass.

From Newton's second law of motion (1.9), the acceleration and acceleration vector of an object, y, due to the gravitational field of an object, x, are given by:

$$a(t) = \frac{Gm_x}{\left\|\overrightarrow{d(t)}\right\|^2} \quad (1.18)$$

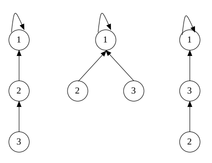
$$\overrightarrow{a(t)} = \overrightarrow{d(t)} \frac{Gm_x}{\left\|\overrightarrow{d(t)}\right\|^3} \quad (1.19)$$

Because the acceleration function is a complex differential equation, to find the position function of a given object under the influence of one or more gravitational fields is not an easy task . Instead, the movement of an object is approximated by using the acceleration vector given by (1.19) in the movement equations (1.14) and (1.15). Therefore, the movement equations of an object y under the influence of the gravitational field of an object x are:

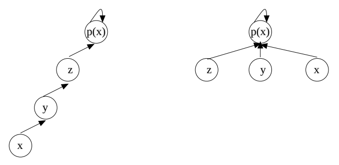
$$v(t + \triangle(t)) = v(t) + \overrightarrow{d} \frac{Gm_x}{\left\|\overrightarrow{d}\right\|^3} \triangle(t) \quad (1.20)$$

$$y(t + \triangle(t)) = y(t) + v(t)\triangle(t) + \overrightarrow{d(t)} \frac{Gm_x \triangle(t)^2}{2\left\|\overrightarrow{d(t)}\right\|^3} \quad (1.21)$$

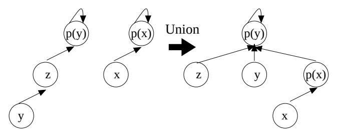
1.3 Optimal Disjoint Set Union-Find Structure A disjoint set Union-Find structure is a structure that supports the following three operators [4]:



(a) Representation of a set with canonical element and tree structure



(b) Path compression in the Find operator



(c) Union operation

Figure 3: Optimal disjoint set union-find structure

- MAKESET(x): Create a new set containing the single element x
- UNION(x, y): Replace the two sets containing x and y by their union.
- FIND(x): Return the name of the set containing the element x.

In the optimal disjoint set union-find structure, each set is represented by a tree where the root of the tree is the canonical element of the set, and each child node has a pointer to the parent node (the root node points to itself) [4]. Figure 3.a shows all the possible trees for representing the set $\{1, 2, 3\}$ with canonical element 1. Basically, the three operations in the optimal structure work as follows:

- The MAKESET(x) creates a set that points to itself
- The FIND(x) returns the canonical element of the set containing x by traversing the tree using the parent pointer, and compresses the path between the node x and the canonical element of the set by setting the canonical element as parent node of each element in the path from x to the canonical element. Figure 3.b shows the path compression performed by the FIND operator.
- The UNION(x, y) operator applies the path compression over the two set trees and makes the canonical element of the set with lower rank to point to the canonical element of the other set. The information about the set rank is stored in the canonical element of the new set. Figure 3.c shows the execution of the UNION operator.

It has been proved that the time complexity of applying any sequence of m UNION and FIND operations on n elements is at most $O\left((m+n)\,log^*n\right)$, where log^*m is the inverse function of the Ackerman function [4]. If $n \leq 2^{2^{2^{2}}} \approx 10^{19728}$ then $log^*n \leq 5$. Therefore the time complexity of any sequence of m UNION and FIND operations on n elements is at most $O\left(m+n\right)$ in practice.

2 Proposed Approach

We developed a robust clustering technique based on the gravitational law and Newton's second motion law. In this way, for an n-dimensional data set with N data points, each data point is considered as an object in the n-dimensional space with mass equal to 1. Each point in the data set is moved according to a simplified version of Equation (1.21). The basic ideas behind applying the gravitational law are:

1. A data point in some cluster exerts a higher gravitational force on a data point in the same cluster than on a data point that is not in the cluster. Then, points in a cluster move in the direction of the center of the cluster. In this way, the proposed technique will determine automatically the clusters in the data set.
2. If some point is a noise point, i.e., does not belong to any cluster, then the gravitational force exerted on it from other points is so small that the point is almost immobile. Therefore, noise points will not be assigned to any cluster.

The simplified version of Equation (1.21) used for moving a data point according to the gravitational field generated by another point (y) is:

$$x(t+1) = x(t) + \overrightarrow{d}\frac{G}{\left\|\overrightarrow{d}\right\|^{3}} \qquad (2.22)$$

where, $\overrightarrow{d} = \overrightarrow{y} - \overrightarrow{x}$, and the constant G includes the division by 2 defined in 1.21.

We considered the velocity at any time, $v(t)$, as the zero vector and $\triangle(t) = 1$. In this way the algorithm does not need extra memory for storing the velocity vector of each data point. Also the algorithm is faster because the number of operations is lower than using the velocity vector.

Due to a constant gravitational force, after a huge (infinite) number of iterations, all the points will be moved to the same position (big crunch), therefore defining a single cluster. We eliminate this effect by reducing the gravitational constant G each iteration in a constant proportion (decay term: $\triangle(G)$). The proposed clustering algorithm is:

```
GRAVITATIONAL( $x$, G, $\triangle(G)$, M, $\varepsilon$)
1  for i=1 to N do
2    MAKE($i$)
3  for i=1 to M do
4    for j=1 to N do
5      $k$ = random point index such that $k \neq j$
6      MOVE( $x_j$ , $x_k$ )  (see Eq (2.22))
7      if dist$^2$( $x_j$ , $x_k$ ) $\leq\varepsilon$ then UNION( $j$, $k$)
8    G = (1-$\triangle(G)$)*G
9  for i=1 to N do
10   FIND($i$)
11 return disjoint-sets
```

In each iteration, the algorithm creates the clusters by using the optimal disjoint set union-find structure and the distance between objects (after simulating the movement of the object by applying the gravitational force). When two points are merged, both of them are kept in the system while the associated set structure is modified. In order to determine the new position of each data point, the proposed algorithm only selects another data point in a random way and move both of them according to equation 2.22 (MOVE function). The algorithm returns the sets stored in the disjoint set union-find structure. It is possible to implement the disjoint sets structure by using an integer array of size N, where at position i, is stored the index of the canonical object of the set containing the data point i.

Because the previous algorithm assigns every point in the data set (noisy or normal) to one cluster, it is necessary to extract the valid clusters. We used an extra parameter (α) to determine the minimum number of points (percentage of the training data set) that a cluster should include in order to be considered a valid cluster. In this way, we used an additional function GETCLUSTERS that takes the disjoint sets generated by the Gravitational clustering algorithm and returns the collection of clusters that have at least the minimum number of points defined:

```
GETCLUSTERS( clusters, alpha, data )
1 newClusters = $\emptyset$
2 MIN_POINTS = $\alpha N$
3 for i=0 to number of clusters do
4   if size( cluster$_i$ ) $\geq$MIN_POINTS then
5     newClusters = newClusters $\cup$ { cluster$_i$ }
6 return newClusters
```

Table 1: Similarities between the proposed approach and Agglomerative Hierarchical Clustering

Property	Gravitational	Agglom. Hierarchical
initial step	N sets with 1 point	N sets with 1 point
iterative steps	Two sets or more can be joined according to gravitational force and distance	Two sets are joined according to distance

2.1 Time Complexity Analysis The time complexity of the MOVE function is constant (the space dimension is considered constant in this analysis). The function UNION in executed in the inner for loop j (lines 4-7) at most once, then the complexity of the inner loop is bounded by N times the execution of the function UNION. Therefore the time complexity of the inner loop j is $O(N)$ in practice. The same analysis is done for loop in lines 9-10. The time complexity of the first loop (lines 1-2) is $O(N)$ because the time complexity of the MAKE function is constant. The time complexity to get the disjoint sets is $O(N)$ by traversing the array of canonical index and separating the objects according to it. The parameter M defines the number of iterations that the algorithm will be executed. This parameter is independent of the data set and can be considered as constant (in the experiments performed here, was fixed to 500. Therefore, the time complexity of the algorithm is $O(N)$. It is clear that the time complexity of the GETCLUSTERS function is generally such that $O(\#clusters) \ll O(N)$. Then the time complexity of the complete algorithm (GRAVITATIONAL and GETCLUSTERS) is $O(N)$ in practice.

2.2 Comparison to agglomerative hierarchical clustering techniques The Gravitational approach (**G-Algorithm**) can be seen as most similar to agglomerative clustering techniques than to partitioning techniques[12]. Table 1 shows the similarities between the G-Algorithm with the agglomerative hierarchical clustering technique.

The G-Algorithm starts with N different clusters, each one with only one data point, in the same way that agglomerative clustering techniques do. In each iteration, more than two clusters can be joined if two of their points are randomly selected and are close enough (with a distance bounded by ε). In agglomerative hierarchical clustering, two clusters are always joined in each iteration after similarities between all clusters are calculated. The main differences between the proposed approach

Table 2: Differences between the proposed approach and Agglomerative Hierarchical Clustering

Property	Gravitational	Agglom. Hierarchical
Complexity	$O(N)$	$O(N^2 logN)$
Storage cost	$O(N)$	$O(N^2)$
Final result	Depend on G, $\triangle(G)$	one big cluster
Point dynamics	mobile	static

and agglomerative hierarchical clustering techniques are summarized in table 2. Clearly, the G-algorithm has several advantages over agglomerative hierarchical techniques, one of them being the time complexity.

2.3 Comparison with Wright's work Although the proposed algorithm uses the gravitational law in the same way that the Wright's work [22] does, several differences can be established between them. Table 3 summarizes the main differences between them.

3 Experiments and Results

To illustrate the applicability of the gravitational clustering (G), tests were conducted over three different synthetic data sets (Gaussian clusters: elliptic and spherical) and over a real data set. For each data set, we normalized the data points to the n-dimensional hypercube $[0, 1]^n$, according to the maximum and minimum values found in the set. Here, n is the number of attributes.

3.1 Synthetic data sets Tests were performed on the three noisy data sets shown in figure 4. The total number of points, N, the number of noisy points, N_n, and the generating center coordinates for these Gaussian clusters are listed in table 4, respectively. The values for the G-clustering parameters were: number of iterations $M = 500$, initial gravitational force $G = 7 * 10^{-6}$, gravitational force loss $\triangle(G) = 0.01$, and minimum distance $\varepsilon = 10^{-4}$.

Table 4 shows the average and variance of the performance over ten runs of the proposed approach, k-Means and fuzzy k-Means. The number of iterations in the fuzzy k-means and the k-means algorithms were 150 in order to perform a fair comparison between the time expended by the **G**-clustering algorithm and these methods. It is clear that in the first data set (with five clusters), 100 iterations in k-means or fuzzy k-means expend the same or more time than 500 iterations of the **G** algorithm.

As is shown in table 4, the proposed approach is robust to noise and in every case, the result were almost the same (variance values lower than 2), and close to the real centers. On the other hand, k means and fuzzy k means were very sensible to noise and the results varied from one run to the other drastically. In some case the variance was very high (see third center in second data set). Figure 5 shows the clusters obtained by the G-algorithm over the data sets in a sample run. It is clear that the proposed approach is able to find the clusters independently of their shape.

For the purpose of an easy presentation of the G-Algorithm properties, we limited the analysis to the first data set. Additionally, we took the parameter values reported here, and we varied each parameter (keeping the others constant) in order to analyze the sensitivity of the proposed approach to such parameters.

3.1.1 Scalability and Dynamics of the G-Algorithm Figure 6 shows the movement of the data points in a simple run after different number of iterations. Clearly, points inside clusters move in the direction of their cluster centers while noisy points stay in almost the same position, confirming our hypothesis of the cluster agglomeration.

Figure 6.b and 6.c show the concentration of points around the cluster centers after different number of iterations. Due to the randomness in the G-Algorithm, some points inside clusters were not moved in the direction of the cluster centers. According to this behavior, it is possible that the G-algorithm does not require the entire data set to determine the clusters.

In order to determine the scalability of the proposed approach, we ran the G-algorithm using different percentages of the data set (between 1% and 100% with differences of 1%). The portion of the data set used was randomly chosen from the original data set. This process was repeated 50 times and the reported results are the average over these repetitions, (see Figure 9).

According to figure 9, the G-algorithm is able to find the clusters in the data set using only 20% (or more) of the data set. Therefore, it is not necessary to use the entire data set to obtain good results.

3.1.2 Sensitivity to α The G-Algorithm uses the parameter α to extract the valid clusters according to their size. For the experiments performed with the synthetic data sets, we found that $\alpha = 0.03$ (clusters with size equal or bigger than 3% of the full data set) is a good value. Figure 8.a shows the number of clusters for different values of α, while figure 8.b shows the number of points covered by the clusters (included in some cluster with the given size). These values are the average over 50 experiments that were performed.

Table 3: Comparison of the proposed approach with Wrigh's work

Property	**Proposed**	**Wright's** [22]
Complexity	$O(N)$	$O(N^2)$
End Condition	Number of iterations	Only one particle remaining
Particles Number	The same each iteration: N	Variable: according to the merging process
Particles mass	Constant = 1	Variable: according to the merging process
Gravity force	The same for each particle, but decrease with the number of iteration	Different for each particle, depend on the mass of the particle

Table 4: Performance of different clustering algorithm on the synthetic data sets

Set	N	N_n	**Generating centers**	**G-Clustering**		**k-Means**		**fuzzy k-Means**	
				Average	Variance	Average	Variance	Average	Variance
1	2530	626	(50,50)	**(49.7, 48.4)**	(0.16,0.02)	(50.8, 52.5)	(0.06, 0.27)	(51.0, 52.33)	(0.02, 0.0)
			(150,80)	**(151.3,77.8)**	(0.31, 0.30)	(162.9, 70.3)	(2.53, 1.82)	(161, 69.6)	(2.35, 0.10)
			(100,150)	**(100.9, 148.2)**	(0.54, 0.97)	(116.6, 145.1)	(767, 3.71)	(81.7, 165.6)	(67, 176)
			(170,160)	**(170.2, 160.1)**	(0.22, 0.23)	(64.3, 197.4)	(83, 236.7)	(165.2,156.9)	(583, 46)
			(200,200)	**(201.0, 197.6)**	(0.49, 0.58)	(195.4, 193.8)	(4.83, 31.3)	(202.7, 203.3)	(8.53, 22.0)
2	1157	346	(50,50)	**(50.3, 49.5)**	(1.66, 1.17)	(65.8, 67.3)	(3.75, 3.49)	(67.6, 64.9)	(0.0, 0.0)
			(90,90)	**(89.5, 89.1)**	(0.01, 0.05)	(190.3, 84.3)	(291, 125)	(113.5, 169.3)	(0.16, 0.07)
			(150,150)	**(149.2, 147.6)**	(0.17, 0.21)	(124.7, 168)	(939, 166)	(174.3, 136.7)	(0.0, 0.11)
3			(50,50)	**(50.4, 49.2)**	(0.91, 0.01)	(58.3, 59.3)	(89.5, 89.2)	(52.1, 49.6)	(0.0, 0.0)
			(100,100)	**(99.7, 101.5)**	(0.20, 5.98)	(152.6, 88.4)	(1482, 128)	(115.8, 103.1)	(0.0, 0.0)
			(150,210)	**(147.6, 209.7)**	(0.09, 0.20)	(138.5, 204.5)	(5.88, 6.0)	(145.2, 207.6)	(0.0, 0.0)

Figure 8 suggests that heuristic techniques can be applied to determine an appropriate value of this parameter. Because this parameter is only used in the extraction of the "valid" clusters (does not affect the gravitational scheme), this parameter can be used for analyzing the internal structure of the data set after the G-Algorithm is applied. For example, according to figure 8.a the data set 1 has two big clusters, each one with a size of 10% of the data set size. Figure 7 shows the clusters obtained by varying the α parameter in a sample run.

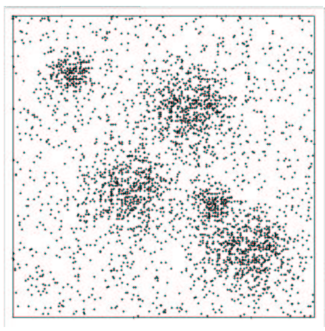
3.1.3 Sensitivity to the Initial Gravitational force (G) G is the most important parameter in the $G-$Algorithm. If it is fixed to a big value, the G-algorithm will form only one cluster (limit behavior). On the other hand, if it is fixed to a very low value, the G-algorithm will not create clusters at all, (see Figure 10). According to figure 10 (using logarithmic scale), the results of the G-Algorithm rely heavily on the value given to this parameter.

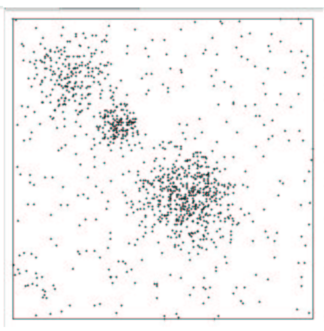
After careful analysis, we could not define a "universal" value for this parameter, i.e., a value that works for all data sets (real or synthetic). Therefore, an optimal value for this parameter depends on the data set, and has to be chosen carefully.

3.1.4 Sensitivity to the Gravitational Force Decay ($\triangle(G)$) $\triangle(G)$ is a very important parameter in the G-algorithm. If it is fixed to a big value, for example, 0.1, the G-algorithm will decrease the gravitational force so fast, that the data points will not have a chance to form clusters at all. On the other hand, if it is fixed to a low value, for example, 0.001, the G-algorithm will decrease the gravitational force so slowly, that the data points will form only one cluster (the limit behavior), see Figure 11. According to figure 11, the G-Algorithm is very sensitive to this parameter. Several experiments inferred that the best value for this parameter is 0.01.

3.1.5 Sensitivity to Merging Distance (ε) We varied the epsilon parameter, ε, between 0.1 and $1e-6$ to determine the sensitivity of the G-algorithm to this parameter. Figure 12 shows that the G-Algorithm is not very sensitive to this parameter. Only when this parameter is set to a big value ($\geq$ 0.01), the results obtained by the G-algorithm become bad (only one cluster). This behavior is expected because a value of 0.01 indicates that two points at a distance of 0.1 are in the same cluster (0.1 is almost 10% of the maximum distance between any two points in the data set).

3.1.6 Suggested parameter values In general, almost all the parameters can be set to a constant value. The only parameter that cannot be set in advance is the initial gravitational force (G). According to our experiments, the appropriate values for the other parameters are: gravitational force decay $\triangle(G) = 0.01$, cluster size $\alpha = 0.03 = 3\%$, and epsilon $\varepsilon = 1e-4$.


(a) Five clusters data set


(b) Three spherical clusters data set

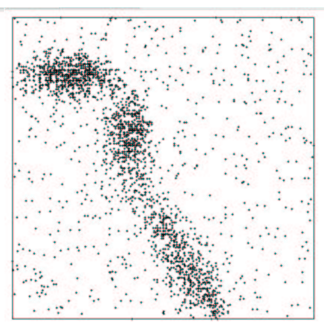
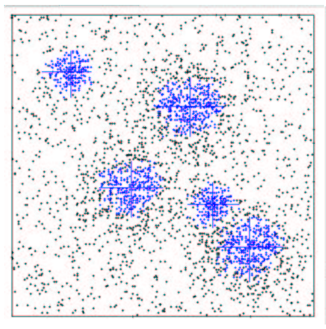
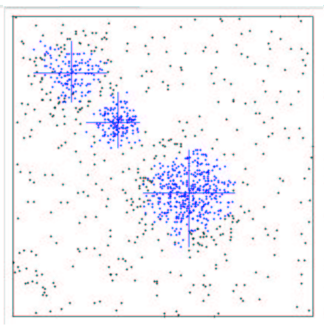
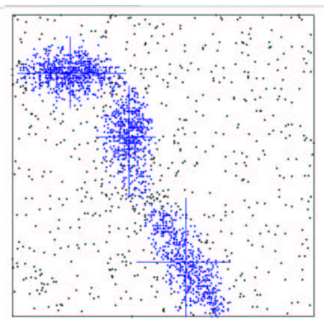
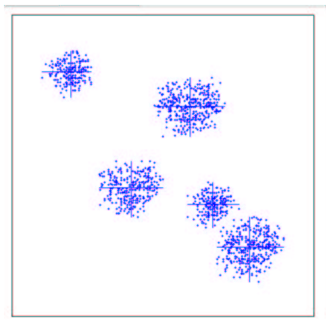

(c) Three clusters elliptical data set

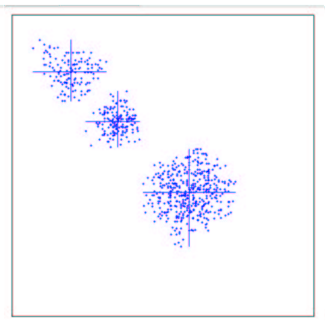
Figure 4: Synthetic data sets


(a) Clusters 'radius' with noise data set 1


(b) Clusters 'radius' with noise data set 2


(c) Cluster 'radius' with noise data set 3


(d) Clusters without noise data set 1


(e) Clusters without noise data set 2

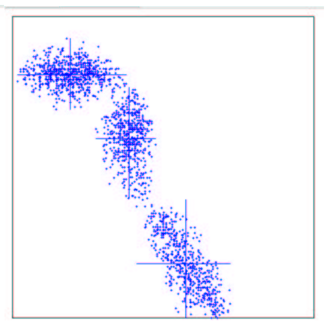
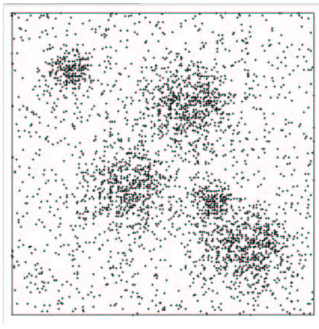
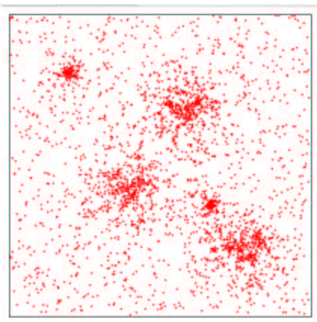

(f) Clusters without noise data set 3

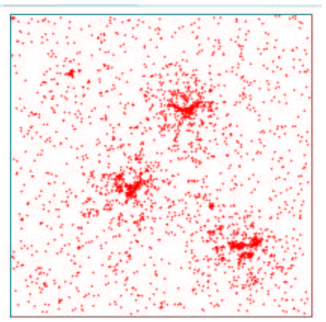
Figure 5: Clusters obtained by the G-Algorithm in a sample run



(a) Original data set

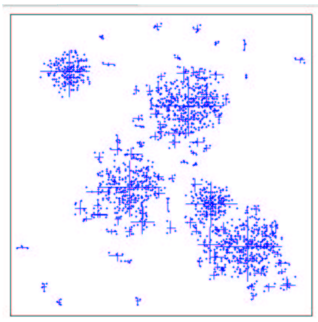


(b) Data points after 100 iterations

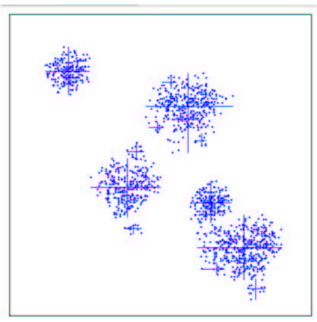


(c) Data points after 200 iterations

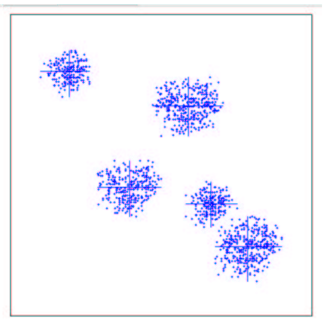
Figure 6: Movement of data points in different iterations



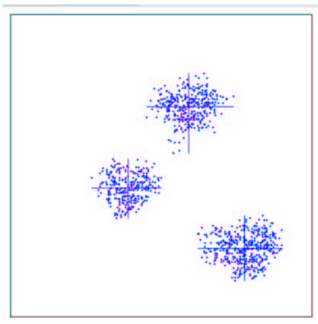
(a) Clusters with at least 3 points ($\alpha = 0.001$)



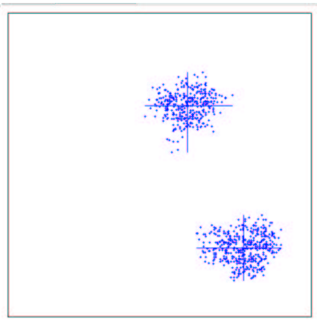
(b) Clusters with at least 10 points ($\alpha = 0.003$)



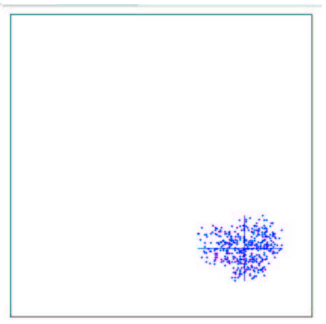
(c) Clusters with at least 100 points ($\alpha = 0.03$)



(d) Clusters with at least 200 points ($\alpha = 0.06$)



(e) Cluster with at least 330 points ($\alpha = 0.1$)



(f) Clusters with at least 360 points ($\alpha = 0.12$)

Figure 7: Resolution levels in the G-Algorithm, as shown in Figure 8.

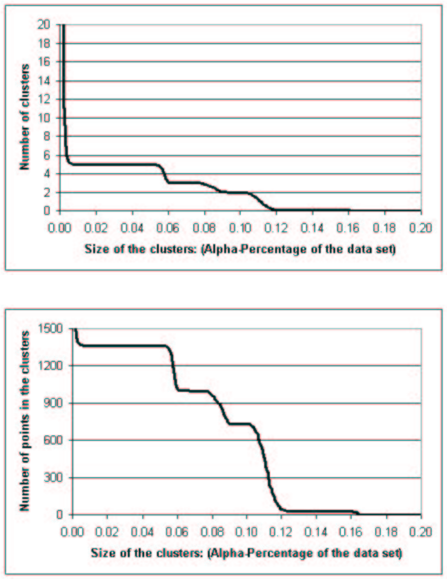


Figure 8: Internal structure of the data set. Number of clusters and total points in the clusters

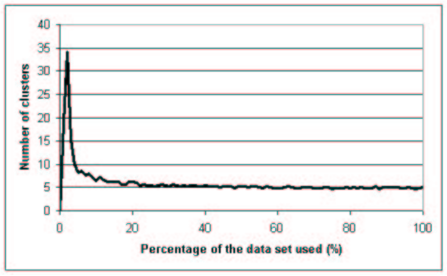


Figure 9: Number of clusters obtained by the G-Algorithm by sampling the data set.

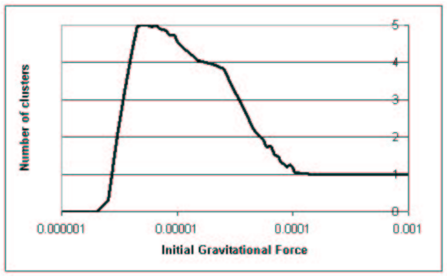


Figure 10: Sensitivity to initial gravitational force (log scale)

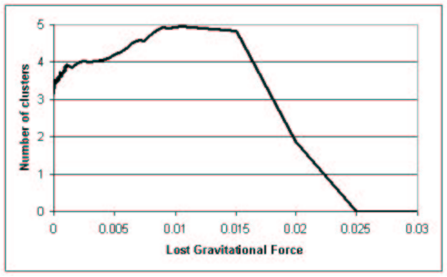


Figure 11: Sensitivity to gravitational force decay

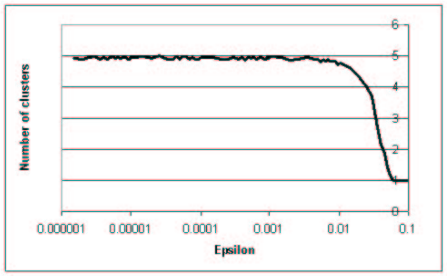


Figure 12: Sensitivity to merging distance (log scale)

3.2 Real Data Set In order to determine the performance of the proposed approach on real data sets, experiments were conducted on the intrusion detection benchmark data set: 10% of the KDD Cup 99 data set, that is available at the University of Irvine Machine Learning repository [2]. This data set is a version of the 1998 DARPA intrusion detection evaluation data set prepared and managed by MIT Lincoln Labs [15]. 42 attributes, that usually characterize network traffic behavior, compose each record of the 10% data set (33 of them numerical). Also, the number of records in the 10% data is large (492021).

In general, the problem of intrusion detection can be stated as follows: to detect when a computer system is being attacked or an intrusion is in progress [1], i.e., to classify a data sample in two classes -normal (there is no intrusion in progress) or abnormal (an intrusion is in progress). Table 5 shows the number of records of each class in the 10% data set.

3.2.1 Preprocessing Because the G-Algorithm can be applied only to numerical data, we generated a reduced version of the 10% data set including only the numerical attributes, i.e., the categorical attributes, including the class label, were removed. Therefore the reduced 10% data set is composed of 33 attributes. Also,

[2]http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

Table 5: Kdd-cup 99 data set

CLASS	NUMBER OF SAMPLES
Normal	95278 (19.3%)
Abnormal	396745 (80.7%)

in order to give the same importance to each attribute, the attributes were normalized between 0 and 1 using the maximum and minimum values obtained.

3.2.2 Experimental Settings Following the analysis done with the synthetic data sets in section 3.1.6, we kept the values for ε, $\triangle(G)$, and α, and we tested several values for the initial gravitational force parameter. The reported results were obtained using the following parameter values: gravitational force, $G = 1e-4$, gravitational force decreasing factor, $\triangle(G) = 0.01$, cluster size, $\alpha = 0.03$, epsilon, $\varepsilon = 1e-6$, and maximum number of iterations $M = 100$. In order to determine the scalability property of the proposed approach, the G-algorithm was run using only 1% (randomly generated) of the reduced data set. This process was repeated 100 times, each run with a different 1%, and the reported results are the average over these repetitions.

3.2.3 Clustering-Classification Score Strategy In order to use the G-algorithm as a classification learning tool, we follow used the following strategy. Let c_i, $i = 1, .., K$ be the collection of K clusters generated by the clustering algorithm. Let T be the number of classes in the classification problem (in our case two), and let $n_{i,t}$ the number of training data records that belong to class t that were assigned to cluster i. This strategy basically consist of two steps:

1. **Building the classification model**: Assign to each cluster c_i the class with more data records assigned to that cluster. Clearly, only the data records used by the clustering algorithm are used (in our case only 1%).
2. **Prediction**: Given an unknown data record, the record is assigned to the closest cluster. The distance between a data record and a cluster is given by the distance between the data record and the center of the cluster.

3.2.4 Results and Analysis There are two elements that define the cost function of an intrusion detection system: the false alarm rate, (**FA**): the system produces an alarm in normal conditions, and the detection rate, (**DR**): the system detects an attack. The best intrusion detection system has low FA and high DR. Table 6 compares the performance of the G-Algorithm against other methods found in the literature.

Table 7: Comparison of the G-Algorithm against $k-means$ and $fuzzy\, k-means$

Algorithm	**FA%**		**DR%**	
	mean	variance	mean	variance
G-Algorithm	**0.1**	**0.07**	**93.15**	**0.38**
k Means	13.3	4.81	94.05	5.47
fuzzy k Means	30.27	100.40	95.29	100.32

As shown in table 6, our approach compares well with such methods. Because we are interested in a simple model, we calculated the average number of clusters generated by the G-Algorithm (3.03). Clearly, the G-algorithm creates a simple model of the data set only using 1% of the data set. Finally, we compared the performance of the G-Algorithm against k-Means and fuzzy k-means. We used $k = 3$ in order to do a fair comparison with the proposed approach. Table 7 shows the results obtained by these three methods over 100 tests.

Although the DR of the k means and fuzzy k means are higher than the G-algorithm the false alarms rates are huge too. Also, the k means and fuzzy k means have a high variance in the performance reached. In general the G-algorithm performs better than (fuzzy) k means.

4 Conclusions and Future Work

A new clustering algorithm based on the gravitational law and Newton's second law of motion was presented in this paper. Several experiments with synthetic data sets and with a real data set were performed in order to show the performance of the proposed approach. The proposed approach successfully determines the number of clusters (unsupervised) in data sets that contain noise (i.e., the algorithm is robust). Also, it is clear that the proposed approach can be used as a pre-processing tool to eliminate the noise from a data set, and as a tool to determine the internal structure of the data set. Although the algorithm is defined by four parameters, according to the experiments performed here, three of them can be set to a constant values (the decay of gravitational force, the clusters size, and the epsilon parameter). Further work will concentrate on:

- Extending the proposed approach to use different metrics (not only the Euclidean distance), and similarities measures (for non-numerical attributes).
- Establishing the relation between the initial gravi-

Table 6: Performance reached by G-Clustering

Algorithm	**Training set size**	**Testing set size**	**FA%**	**DR%**	**Complexity**
G-Clustering	1%	100%	**0.1**	**93.15**	$O(n/100)$
EFRID[10]	80%	20%	7.0	98.95	$O(n)$
RIPPER-AA[7]	80%	20%	2.02	94.26	$O(n\,log^2 n)$
SMARTSIFTER[23]	80%	20%	-	82.0	$O(n^2)$

tational force and the data set, in order to set automatically this parameter. According to figure 10 it is possible to design an heuristic technique for setting this parameter.

- Integrate fuzzy logic in order to perform a better cluster analysis.

5 Acknowledgments

This work was supported by the Defense Advanced Research Projects Agency (F30602-00-2-0514). Nasraoui is supported by an NSF CAREER Award (NSF-IIS-0133948).

References

[1] E. Amoroso, *Intrusion Detection.* Intrusion.net Books, 1999.

[2] G. Beni, and X. Liu, *A least biased fuzzy clustering method.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 16(9):954-960, 1994.

[3] J. C. Bezdek, *Pattern recognition with fuzzy objective function algorithms.* Plenun Press, 1981.

[4] T. Cormer, C. Leiserson, and R. Rivest, R., *Introduction to Algorithms.* McGraw Hill, 1990.

[5] L. Davidson, *Visualizing clustering results.* In Proceedings of the Second SIAM International Conference on Data Mining . 2002.

[6] H. D. Eckhardt, *Kinematic Design of Machines and Mechanisms.* McGraw Hill, 1998.

[7] W. Fan, W. Lee, M. Miller, S. J. Stolfo, and P. K. Chan, *Using artificial anomalies to detect unknown and known network intrusions.* In Proceedings of the first IEEE International conference on Data Mining, 2001.

[8] H. Frigui, and R. Krishnapuram, *A robust clustering algorithm based on the m-estimator.* In Neural, Parallel and Scientific Computations, 1995.

[9] H. Frigui, and R. Krishnapuram, *Clustering by competitive agglomeration.* Pattern Recognition, 30(7):1109-1119, 1997.

[10] J. Gomez, and D. Dasgupta, *Evolving Fuzzy Classifiers for Intrusion Detection.* In Proceedings of the 2002 IEEE Workshop on Information Assurance, June 2002.

[11] F. R. Hampel, E. M. Ronchetti, P. J. Rousseeuw, and W. A. Stahel, *Robust statistics: the approach based on influence functions.* John Wiley & Sons, 1986.

[12] J. Han, and M. Kamber, *Data Mining: Concepts and Techniques.* Morgan Kaufmann, 2000.

[13] J. M. Jolion, P. Meer, and S. Bataouche, *Robust clustering with applications in computer vision.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 13(8):791-802, 1991.

[14] R. Krishnapuram, and J. M. Keller, *A possibilistic approach to clustering.* IEEE Transaction in Fuzzy Systems, 1(2):98-110, 1993.

[15] Lincoln Laboratory MIT. http://www.ll.mi.edu/

[16] J. MacQueen, *Some methods for classification and analysis of multivariate observations.* In Fifth Berkeley Symposium on Mathematics, Statistics, and Probabilities. 1967.

[17] J. B. Marion, and S. T. Thornton, *Classical Dynamics of Particles and Systems.* Brooks/Cole Pub Co., 1995.

[18] O. Nasraoui and R. Krishnapuram, *A Novel Approach to Unsupervised Robust Clustering Using Genetic Niching.* In Proceedings of the Ninth IEEE International Conference on Fuzzy Systems, 170-175, 2000.

[19] P. J. Rousseeuw, and A. M. Leroy, *Robust regression and outlier detection.* John Wiley & Sons, 1987.

[20] K. R. Symon, *Mechanics (3rd Edition).* Addison Wesley, 1971.

[21] K. J. Waldron, and G. L. Kinzel, *Kinematics, Dynamics, and Design of Machinery.* John Wiley & Sons, Inc., 1999.

[22] W. E. Wright, *Gravitational Clustering.* Pattern Recognition, 9:151-166, Pergamon Press, 1977.

[23] K. Yamanishi, J. Takeuchi, and G. Williams, *On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms.* In Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2000.