Clustering with the Levy Walk: "Hunting" for Clusters

Benjamin Schelling
University of Vienna
benjamin.schelling@unvie.ac.at

Claudia Plant
University of Vienna
claudia.plant@univie.ac.at

***Abstract*—The Levy Walk (or Levy flight) is a concept from Biomathematics to describe the hunting–behaviour of many predatory species. It is a very efficient way to find prey in a very short time frame. We now want to use this concept in a clustering–context to – if you so will – "hunt" for clusters. We describe how we convert this concept into an efficient way to find cluster centres by linking the data points through the path the Levy Walk takes. The clusters are then created by statistical analysis of the links between the data points. The result is a Clustering algorithm that works on massive datasets with extremely high level of noise. It is not dependent on any form of cluster shape and is almost free of parameters. The only choice one has to make is the precision of the Clustering, which in turn determines the runtime.**

I. Introduction

As the data we compile from real world and experiments increases, so increases the need for algorithms to automatically interpret these masses of data and reduce them to understandable and interpretable amounts. This should entail the interesting aspects, which might otherwise remain hidden in a sea of noise. The more data we have, the more complicated it gets to understand what we are actually dealing with, hence, we need a way to deal with large amounts of data, especially if they contain lots of noise. This noise is usually not uniformly distributed, but scattered over the complete space. Furthermore there are local variations and noise is more prevalent in some places than others. To handle these landscapes of noise we introduce a novel algorithm. It is based on the Levy Walk, the mathematical description of the hunting behaviour of many predators ranging from simple protozoan to full-grown mammalian hunters. We take the similarity in complexity between a sufficiently extended and complicated dataset and a Petri dish filled with nutrient solution as a starting point to convert this model of behaviour from hunting for nourishment to hunting for clusters.[1]

A. Contributions

We focus on clustering problems with extreme levels of noise in massive data sets, where clusters might be the exception rather than the norm. For this we have created an algorithm based on the Levy Walk with the following features:

1) The algorithm is capable of finding *clusters of arbitrary shape* in data even if the data contains *enormous levels of noise*, and even if such noise is distributed in a quite disadvantageous way. We show that the algorithm is, at least on some datasets, superior to time-tested and state-of-the-art techniques.
2) The algorithm can do this (almost) *without parameters*. It does have a runtime/precision-parameter though, which can be freely selected, considering certain constraints.
3) While the algorithm is non-deterministic, which is often considered a disadvantage, we have found a way to make use of this fact by combining multiple clusterings.

[1] Additional Information: https://cs.univie.ac.at/dm/downloads/

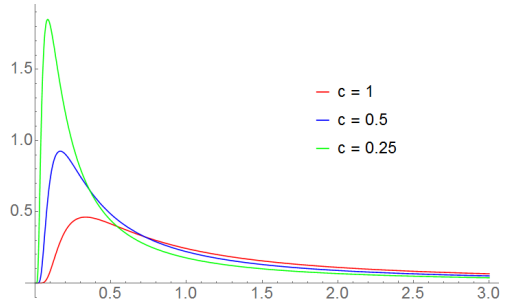
II. Related Work

Because of the size of some data sets, algorithms have been created to deal with them automatically. Automatically may mean setting some parameters first in some algorithms, which is not always an easy decision to make. Some of those algorithms like SYNC [1] can handle the clustering process fully automatic, while others, like DBSCAN [8] rely on correctly chosen parameters. Commonly clustering algorithms belong to one of two groups:

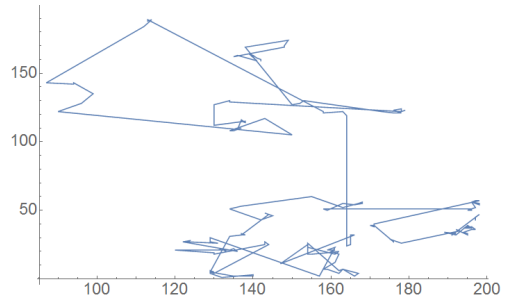
PDF-based Clustering: The common ground behind these group of clusters is the idea that a Probability Density Function (PDF) can describe the data. The drawback here of course is that if the data does not adhere to the assumptions, the clustering will most likely fail. Most prominent representative for this group are k-Means [9] and EM [2].

Density-Based Clustering: Algorithms of this kind, like SYNC, look for areas of high density in the data. This causes them to be independent of any assumption on the distribution of the data, which makes it possible to find clusters of various shapes. Many methods of this group, like DBSCAN [8] have the advantage that they are capable of dealing with noise. Related to this group is the concept of Spectral Clustering.

The Levy Walk is – as far as we know – a completely new concept in the context of clustering. Its only use (to the best of our knowledge) in an area of computer science is its use as part of a search algorithm known as Cuckoo Search (CS) [7]. CS is a member of a family of animal-themed search algorithms, based on swarm intelligence methods, like Ant Colony Optimization [3] or Honey Bee Algorithm [6].



(a) The Levy Distribution.



(b) How the Levy Walk actually looks like.

Fig. 1. The Levy Distribution.

III. The Levy Walk

The Levy Walk is a description for the behaviour of hunting animals which often describes them very well. It is found from tiny predators like e.coli, hunting for nutrient solution in a petri dish, to sharks hunting for fishes. In this paper one may imagine the predator closer to an e.coli bacterium than to a shark, which has naturally a somewhat more complicated hunting pattern than e.coli. Nevertheless, the basic aspects are the same. The Levy Walk is distinguished by its characteristic feature of roaming the whole of its realm and its ability to find the areas with the highest densities of prey efficiently. After all, billions of years of evolution have shaped these hunting patterns.

The Levy Walk is based on the Levy–Distribution, about which we need to present a few basic facts. The probability density function of the Levy Distribution is given by

$$f(x;c) = \sqrt{\frac{c}{2\pi}} \quad \frac{e^{-\frac{c}{2x}}}{x^{3/2}} \tag{1}$$

As we see in Fig. 1a the main mass of the distribution is close to zero, but the Levy distribution is a heavy-tail distribution and its expected value is infinite. If we draw a Levy-distributed random number we mostly get small numbers, with a decent chance of bigger numbers and sometimes enormous ones. If we have a scaling parameter of 1, we get a number bigger than 1000 in more than 2.5% of the cases.

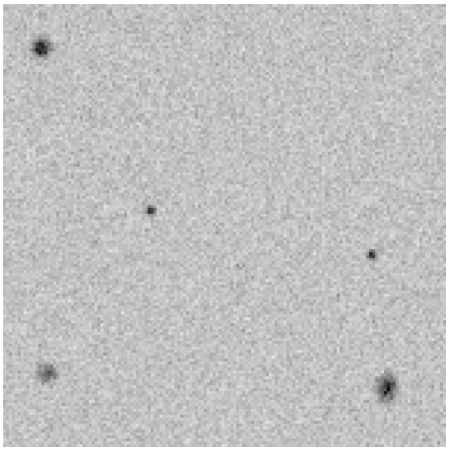
This leads to the special characteristics of the Levy Walk. Assume that we are in a two-dimensional area with a certain amount of unevenly distributed ”prey”. The predator starts at a random point in the area and assesses the current situation. It can see a small bit of the surrounding area. A predator is always on the hunt, which means that it will always ”jump” from one place to another in the hope of finding more. If the current amount of prey in its small surrounding area is high, it is more likely to make only a small jump, but could still make a giant leap. If the current amount is rather disappointing, it will most likely take a big leap, but might still take a small jump. As the predator can see a bit of its surrounding area, it can guess in which direction there might be a higher amount of prey and it will move in this direction. It can not stop or change direction during the jump.

In this context this means the following: The amount of prey – data points – in the small area we are currently in, determines the scaling parameter c of the Levy distribution. We draw a random number from it and this sets the distance the predator will jump. The direction is set by the gradient of the prey-concentration in the current area. If the gradient is zero, e.g. there is no prey around, the direction is randomly chosen. We land in a new place in the area and repeat the process. If we land outside the domain, we just redo the jump. We remember every place we ever landed on and from this information we hope to construct the clusters. The basic idea is that the Levy Walk will spend more time/jumps in areas where clusters are and will jump inside a cluster significantly more often than outside of it.

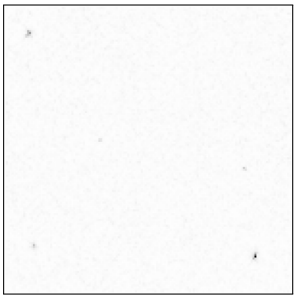
For now we will let all of this happen in a grid, to show that the algorithm actually works and to enable comprehension. While Levy Walk Clustering is not depending on a grid, it is useful to get it started. This way we can easily show the potential and efficiency of this new approach, but we are very aware of the drawbacks of a grid and will get rid of it in future works.

We created an artificial dataset with about 585.000 data points in noise and 7.000 data points in various clusters on a 150x150 grid and let the Levy Walk jump 1.000.000 times. We see in Fig. 2b, how often the Levy Walk is in various grid cells of the dataset. It becomes obvious that the Levy Walk gravitates to the centre of a more pronounced cluster. We can also show that the Levy Walk does not ignore the other clusters. If we plot only those grid cells, where the Levy Walk is five times as often as compared to the average, we get Fig. 2c. We see that the Levy Walk finds all the clusters and even a bit of their shape.

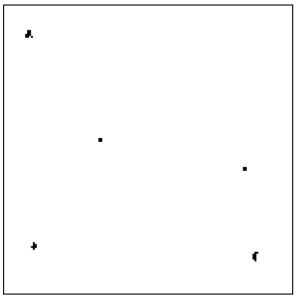
One strength of the Levy Walk is that it stays rather unfazed in the company of excessive noise. We have seen that a noise-level of more than 95% does not pose too much of a problem for it. Now let us add even more noise, but not just noise – let us make it nasty noise. We take the same set of clusters and noise and add another 870.000 noise points, but this time in a gaussian shape. See Fig. 3a for a clearer picture. We now have a noise level of more than 1.455.000 points with the same 7.000 points in clusters. It is understandable that the Levy Walk now has more difficulties than before. We let the Levy Walk jump 1.000.000 times like before and plot the same ”distillate” as before—Fig. 3b. There are now a few black spots left in areas where no clusters are in the data. We see that a threshold of jumps five times above the average is too



(a) The dataset.

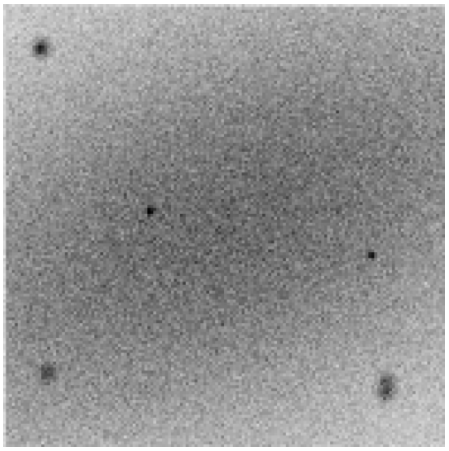


(b) The Levy Walk on the dataset.

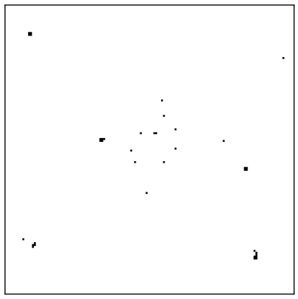


(c) The distillate of the Levy Walk.

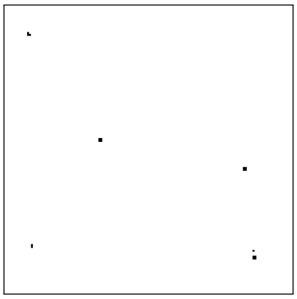
Fig. 2. Dataset: 585.000 noise points, with a few clusters with sizes between 500 and 2500 points thrown in. Since the data set is so massive we can only plot the grid cells and not the data points. The darker a grid cell is, the more points are contained in it.



(a) The new dataset.



(b) The distillate of the Levy Walk.



(c) The distillate of the Levy Walk – cut 7.

Fig. 3. Dataset: More than 1.455.000 noise points and the same clusters as before.

low in this case. We therefore raise it to 7 times the average and get Fig. 3c. The "noise" is gone. We see that the Levy Walk Cluster algorithm is pretty good at finding clusters—or rather cluster centres. We still need to find a way to know how high we have to set the barrier to distinguish between noise and cluster points though.

For now, we have found a way to get the most pronounced points of a cluster. However, we have not yet discussed a way to identify those points as belonging to a particular cluster and finding other points that belong to the same cluster.

TABLE I
ABBREVIATIONS USED THROUGHOUT THE PAPER

Abbreviation	Meaning
D	data points
n	number of data points
d	dimension of data points
G	$d-dim$. Grid containing the data points
k	number of grid cells per dimension
L	Levy Walk executed on G
m	number of steps L takes
C	cluster in D
l	number of clusters in D

IV. Making Clusters – The centres

Let us assume we have n data points on a grid G. They make up the whole of the data D, $|D| = n$. The Grid is of the form $G = I^d$ with $I = \{1, 2, \ldots, k\}$, d the dimension of the data, $|G| = k^d$. There is no need for the Grid to have length k in every dimension, but we lose unnecessary variables this way and if G were of not identical length, we could easily rescale. All the data points now fall into one of the grid cells. On this grid we let the Levy Walk L loose. For all in all m jumps it moves from one to cell to another, hence we get $L = \{x_1, x_2, \ldots, x_m\}$ with the x_i, $i \in \{1, 2, \ldots, m\}$ representing one of the grid cells. We assume the data D to contain l clusters C_i, $i \in \{1, 2, \ldots, l\}$, of various shape and size. We now want to find as many of those clusters as possible and those grid cells x_i that belong to them.

Definition 1 (Cluster centres): We will define those grid cells as **Cluster Centres** which are hit by the Levy Walk significantly more often as a grid cell – which is not part of a cluster – can be expected to.

We will show in the next few paragraphs how "significantly" is understood here and derive Equation (2), which gives us the threshold between cluster centres and regular grid cells. This threshold t is dependant on the number of steps m, the grid size k and the "confidence interval" $x \cdot \sigma$. We get to it in the next few paragraphs.

$$t(m,k) = \exp[\log[\frac{m}{k^d}] + x \cdot \sigma] \quad (2)$$

So, how do we get to this equation? L consists of a succession of x_is. When there is only noise, all grid cells will be similarly often visited, because there is no area where the Levy Walk finds a cluster/reason to stay. Of course one grid cell will be by default the grid cell where the Levy Walk hits most often, but because its environment is (approximately) uniformly distributed, the gradient of its environment will not necessarily point towards it. If, on the other hand, there are clusters, then the most often visited grid cell will most likely be the centre of one of the clusters and the gradient of its environment will mostly point towards it. Therefore, the Levy Walk will much earlier find it and when it deviates a bit from the centre there is a decent chance that it will go back there. From all this follows, that a cluster centre will have a significantly higher visiting rate, than a data point in the barren wasteland, that is a dataset without clusters.

Let us put this in mathematical writing: Assume the dataset contains no clusters. We have m entries in L which will get distributed through the Levy Walk over k^d grid cells. We can now expect for the average grid cell to contain roughly $\frac{m}{k^d}$ entries. This is the expected value for a grid cell, if all grid cells are the same. But it is not quite that simple. The grid cells have a slightly different probability to be hit or rather if they are hit in some cases the Levy Walk might be reluctant to leave. Like we mentioned before the amount of data points in a grid cell determines the scaling parameter of the Levy Walk. The scaling parameter is somewhere between 0.01 and 1. We can not let the parameter be 0 or else the jump lengths would be 0 too and the Levy Walk would not walk any more, but decide to live the rest of his life wherever he currently is. The Maximum of 1 on the other hand is somewhat randomly chosen. We could just as well use any other number bigger than our minimum, but the higher the maximum is, the more often we have to redo the jump, because it would jump outside of the grid. The maximum also determines – to a certain extent – how "local" the Levy Walk is. A smaller maximum would mean that close cells are more often connected than they would be otherwise and hence we should not choose it too small.

If we are at a grid cell with a small scaling parameter, the Levy Walk has a rather high probability to stay in this grid cell. This follows from the Levy-distribution-density (see Fig. 1a). The Levy distribution has its peak before 0.5 and decreases strictly monotonously from there on. Most of the randomly drawn numbers will therefore be rather small and the Levy Walk will not move by much and will most likely not leave the grid cell it currently is in. From all this follows that the Levy Walk will stay far more often in some cells than in others. This is corroborated, when we look at Fig. 4a. Its plotted there how often the Levy Walk stays in a grid cells and it becomes clear that some are simply more favoured than others.

We can not really do much with this density-histogram, but if we take the logarithm of all these values we get Fig. 4b, which looks very much as if its normal-distributed. The Kolmogorov-Smirnov-test gives a positive value for this hypothesis and the relation between mean and variance between the logarithmic version and the original one is as expected. We can therefore see the original density-distribution as log-normal distributed.

We can work with this: We know that the expected mean for the density is $\frac{m}{k^d}$, and therefore $Log[\frac{m}{k^d}]$ for the logarithm of the densities. The variance is harder to calculate, but we do not quite need to. We could simulate it, by letting the Levy Walk jump for as long as we want it to in a grid of our choosing, with a uniformly distributed dataset. From this we get σ^2, the variance we are looking for in the normal distribution.

With the mean μ and variance σ^2 of the distribution found, we could calculate the maximum/threshold we are looking for. If we take a threshold of $\mu + x \cdot \sigma$, then we could assume that $\frac{1}{2}\operatorname{erfc}(\frac{\mu - x}{\sqrt{2}\sigma})\%$ of the grid cell values lie in the interval $[0, \mu + x \cdot \sigma]$. For $x = 5$ this would mean that 0.999999426% of all grid cell values lie in this interval. This would be quite enough for $|G| = 1.000.000$, but one might want to increase it, if $|G|$ were bigger. The grid cell values are LogNormal-distributed, hence we get a threshold of $Exp[Log[\frac{m}{k^d}] + x \cdot \sigma]$, with x depending on $|G|$ and σ not yet determined.

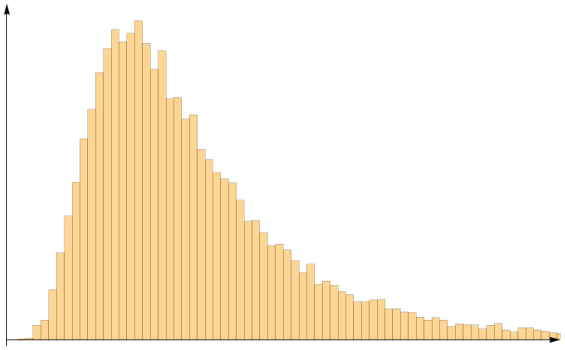
We have shown how we get to Equation (2) and we can state:

Conclusion 2: Grid cells $x_i \in G$, with G the grid over the data D, are **Cluster Centres**, as defined in definition 1, if the Levy Walk has been in x_i more often as the threshold in Equation (2) deems likely.

But basically, these calculations are not necessary. When we look at Fig. 4c we see that the maximal grid cell values differ wildly, depending on if there are cluster present in the data or not. Therefore, we do not need very precise calculations to compute the threshold between cluster-free data and data with clusters. We have implemented in the code a simulation that lets the Levy Walk jump on a grid on uniformly distributed data without cluster a few times according to the chosen setting. We look at the maximal values for grid cells and create the threshold from that. This way we can compute the threshold precisely enough for this prototype and do not have to have the earlier considerations completed.

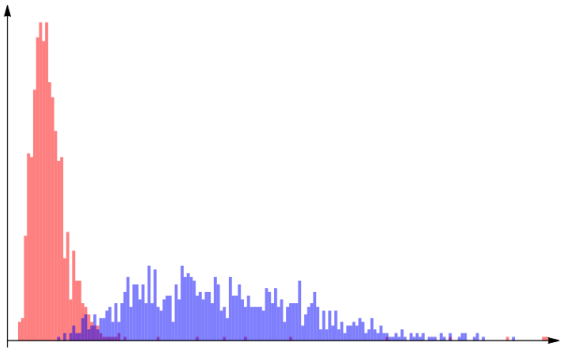
V. Making Clusters – The Clusters themselves

We have now found a way to get a cluster centre. We basically test if any grid cells are above the threshold stated in Equation (2) and if there are we take the one with the most hits. Now how do we get from a cluster centre to a real cluster? We have some information that we have not used at all until now. When we look at L it is basically a progression of links between grid cells. Whenever it jumps from one to another the Levy Walk says that they are close. If they are close enough – the Levy Walk jumps very often from one to the other – then they probably belong to each other/are part of the same cluster.



(a) Histogram of the densities in the gridcells for the Levy Walk, when we had it jump for 1.000.000 times.

(b) Logarithmic Histogram of Fig. 4a, with the Graph of the Normal Distribution superimposed on it.



(c) Histogram of the maxima for the whole grid. Red are the maxima if there are no cluster present, blue if there are. 1000 test-runs each. Red has a mean-value of 873, blue of 3712. $m = 1.000.000$.

Fig. 4. The behaviour of the Levy Walk and its maxima.

We have in L the sequences x_i, x_j whenever the Levy Walk jumps from grid cell x_i to grid cell x_j. On a dataset without clusters, which only contains noise, these sequences will have a certain frequency depending on variables like k and m. Our assumption is that in a dataset with clusters in it, these jumps will occur with a much higher frequency, than without clusters. This seems very likely considering Fig. 2 and 3. A cluster centre functions as a sort of attractor. The gradient of close cells often point to it and they therefore have a very high chance of jumping to it. All these cells are then labelled part of the cluster and we check which points now jumps towards them (or other cluster cells) and add them too. We repeat this step then as long as the cluster grows. When we have reached its final size, stop, declare all the cells as a cluster and cut it out of the dataset and the Levy Walk. The main question is how many of these jumps have to happen for it to be close enough?

If we were to assume the jumps as random, then the jump x_i, x_j would have probability $\frac{1}{k^{2d}}$ and hence a frequency of $\frac{m}{k^{2d}}$. But it is far more likely for a jump to happen when the grid cells are close and it is absolutely necessary for the gradient to point in its direction. If they are directly next to each other, then the probability would be highest when the gradient points into the middle of it. Depending on the scaling parameter of the Levy Walk it would be somewhere in the range $[0.0473882, 0.256917]$. So up to 25% of the possible jumps would hit its "target". We can expect $\frac{m}{k^d}$ jumps to start in the first cell and $\frac{m}{4k^d}$ to hit its direct neighbour, at least if the scaling parameter is very close to 1. If the scaling parameter is smaller than that, it will lie below $\frac{m}{4k^d}$. The trouble is that we do not know how often the scaling parameters appear in the dataset without looking at it, so we have to assume that all of them are equally likely. If we do that and integrate over all the possible scaling parameters between $[0, 1]$ to get the average value we get ≈ 0.2187. This means that we can expect $\approx 21.87\%$ of the jumps to reach its neighbouring cell.

One problem though lies in the fact that, when the cluster grows, we will have to raise this threshold. After all it is far more likely to hit 5 cells, than 1. The most accurate solution to this problem would be to calculate the probability for each cell to hit the cluster. This probability would depend on the cells scaling parameter, its size, the position of the cluster relative to the cell at hand, the amount of jumps and the number of grid cells. But this would be quite tedious, so to shorten this we will take the threshold of $\frac{m \cdot 0.2187}{k^d}$ as a starting point and increase it depending solely on the size of the cluster. We have decided to use a threshold of

$$t(m, k, p) = \frac{m \cdot 0.2187}{k^d} \cdot (2 - \frac{1000}{1000 + p}) \tag{3}$$

where p stands for the size of the cluster. If $p = 0$ we would have $\frac{m \cdot 0.2187}{k^d}$ as the threshold and it would eventually rise up to $\frac{2 \cdot m \cdot 0.2187}{k^d}$, if the cluster were to grow on and on.

VI. Runtime and Pseudocode

Reminder: We have n data points on a grid $G = I^d = \{1, 2, \ldots, k\}^d$. On this grid we have the Levy Walk $L = \{x_1, x_2, \ldots, x_m\}$ with x_i, $i \in \{1, 2, \ldots, m\}$ and l clusters C_i, $i \in \{1, 2, \ldots, l\}$.

Now how do we proceed exactly? First we fit the data into the grid G, then we let the Levy Walk jump around on the grid, following the strategy we explained earlier. We then check if there are any cluster centres (see Section IV) and if there are, we find the grid cells that are part of the cluster (see Section V). When we have found the whole of the extension of the cluster we cut it out. This means that we delete the data points that are part of the cluster from the grid and save them in a

Algorithm 1 Levy Walk Clustering

Require: Data D, Gridsize k

procedure LEVY(D, k)
  Initialize: $I^d \leftarrow D$ ▷ $\mathcal{O}(n)$
  Levy Walk: Let L on I^d ▷ $\approx \mathcal{O}(n)$
  while $C_i \in G$ **do** ▷ l times
    while C_i grows **do** ▷ $|C_i|$ times
      if $(x_i, y) \in L > threshold$, $y \in C_i$ **then**
        $C_i \leftarrow x_i$ ▷ $\mathcal{O}(n)$
      end if
    end while
    Cut Cluster out ▷ $\mathcal{O}(n)$
    Redo Levy Walk ▷ $\mathcal{O}(n)$
  end while
  return $C_1, \ldots, C_l$
end procedure

separate structure. We then redo the Levy Walk. After that we check if we find another cluster centre and repeat the process if necessary. If not we return our clusters, if any were found, and terminate.

Following the pseudo-code we get a runtime expectancy of roughly $\mathcal{O}(n) + \mathcal{O}(n) + l \cdot [|C_i| \cdot \mathcal{O}(n) + \mathcal{O}(n) + \mathcal{O}(n)] \approx \sum_{i=1}^{l} |C_i| O(n)$.

It is not immediately obvious, but the runtime depends heavily on the chosen k. We can explain it this way: After we took a look at the data points and fit the grid with size k^d over it, we have to let the Levy Walk jump. Now, the number of jumps the Levy Walk should take depends directly on k. When we have a grid, it is not relevant how many data points are present (see Section VII for details), only that the Levy Walk jumps often enough on it, so that the connections between grid cells become significant enough to create clusters from them. Therefore, $|L|$ has to exceed a certain level depending on $|G|$ and therefore on k. k can be (mostly) chosen freely, but there is something like an upper and lower bound for it, as we will see in Section VII. For one thing k^d should not exceed $\frac{n}{10}$ on danger of the Levy Walk losing its meaning. We see $|L|$ is directly related to k and the size of L heavily influences the runtime for creating clusters, because creating clusters is mainly a task of counting the sequences of $(x_i, y), y \in C_j$ in L. The shorter L is, the faster this can be done, especially considering that there will be fewer x_is, because the grid is smaller. The size of C_i also depends on k. If the grid cells are bigger, than we need less of them to completely cover C_i and $|C_i|$ is a relevant term in our earlier runtime-calculations.

Therefore holds: The smaller we choose k, the faster the algorithm will be. But this carries the risk of loosing some of the precision Levy Walk Clustering could have. See Section XI for details on this.

VII. Deciding the details

There are some decisions that have to be made, for the Clustering to be at its most effective. In this section we wish to address these difficulties and show how they can be dealt with, if they have to be dealt with at all.

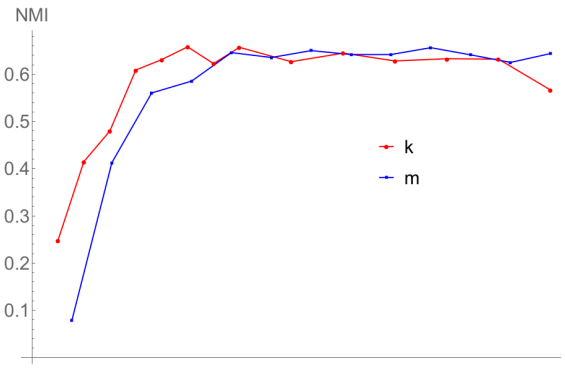


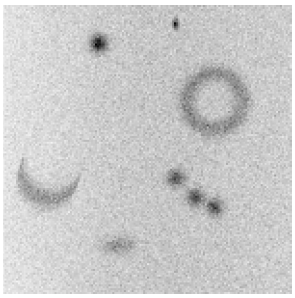
Fig. 5. Quality of Clustering relative to various parameters. m is in the range from 300.000 to 1.000.000 and k ranges from 30 to 200. We see that the quality of clusterings is very similar if the parameter are above a certain threshold. Hence, we know how to set them. Data set shown in Fig. 10a

The size of the Levy Walk L:. We ran our algorithm on the same dataset with different settings for m multiple times and average the results (Fig. 5). It becomes clear from this, that we only need to pass a certain threshold for the Clustering to reach its maximal reachable quality. On the dataset with 22.500 grid cells, this threshold lies at approximately 500.000 jumps. But, if we take a look at some two-dimensional clusterings we see, that this is due to some statistical fluctuations. These cause some rather remote parts of the cluster to be recognized as part of it, but lose some areas closer to the centre which should be found. This is less likely for higher values of m and therefore 800.000 seems to be a wiser choice here. From this follows that $\frac{800000 \cdot |G|}{225000} \approx 35|G|$ is a valid choice for the size of L.

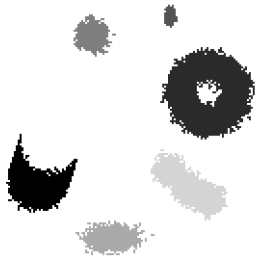
The size of the Grid G: While m is a parameter that – in the end – has little effect on the quality of the clustering, the size of k is a very significant decision, that determines the quality of the clustering as well as the runtime. We nevertheless feel the need to stress that a "wrong" decision – e.g. a decision that leads to false results – is hard to come by. We see in Fig. 5 that k can be chosen in a very broad range of values. Nevertheless, (very) small k have two serious disadvantages: 1) The precision of the clustering has to suffer by default, due to the imprecise excision of a cluster. The smaller k the coarser the grid will be and hence every mistake in computing the form of a Cluster will lead to a far bigger drop in NMI, than if k were bigger. 2) The Grid needs to have a certain size in every dimension because the Levy Walk depends on the possibility of taking very large steps. If k would be chosen too small, the Levy Walk would almost become a random walk and lose the purposiveness of the Levy Walk, developed and shaped by evolution.

VIII. Using the Randomness: Adding multiple Clusterings up

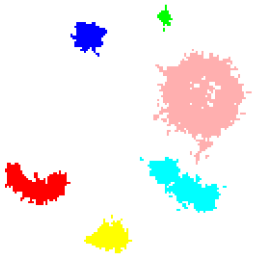
Usually it is considered a disadvantage if a clustering algorithm is non-deterministic. But there is also an advantage to it, which we will now discuss. We have a data set (see



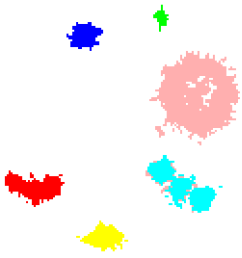
(a) The dataset: Three Gaussian Clusters, a Half-moon, a ring and one cluster consisting of three gaussian sub-cluster.



(b) The ideal Clustering.



(c) The combination of the clusterings.



(d) The combination of the clusterings, using the irregular clusterings.

Fig. 6. Simulation results for the technique described in VIII.

Fig. 6a) and various clusterings of it. Each of these clustering has a certain quality, which means that some clusterings are further or closer to the "true" clustering in certain areas. Some clusterings will have a very good approximation of one cluster, but only a part of another cluster and maybe even a bit too much of a third one. Now, if we assume that Levy Walk Clustering is a valid way of clustering, then we can say that most of the information we have is true, but maybe not all of it. We will now take those various clustering and assemble the information they each have into a single clustering. Through this process we hope to eliminate some of the false information and add up information that actually belongs together. In a certain way we are adding an ensemble-clustering approach for the Levy Walk Clustering, which uses the fact that Levy Walk Clustering is non-deterministic.

Let us say we have j clusterings $Clu_i, (i \in \{1, 2, \ldots, j\})$, of a dataset. If a grid cell is only in one clustering part of a cluster but only considered as noise in the others, then we can assume that it is noise over all and the error lies in the single clustering. Those cells which appear in less than a certain percentage of the clusterings – we propose 30% for this – will therefore be labelled as "noise cells". If a grid cell is in more than a certain amount of the clusterings part of a cluster than it is very unlikely that it is noise. For this we propose 80%, but it does not seem to be too important how exactly this value has been chosen, just like the other value. We will call these cells "core cells".

We start with one of those "core cells". We take all the clusters of the various clusterings that contain this "core cell" and add them together, excluding those cells that are considered "noise cells". We have a new cluster now, which might contain other "core cells". We add the cluster which contains these new "core cells", again excluding the "noise cells". This way we should obtain a new cluster which should be closer to the "ideal" cluster which we strive for. Let us see how this actually pans out.

We have chosen an artificial dataset with clusters varying widely in size, shape and distinctness (see Fig. 6a) and created 8 clusterings for this dataset. We used the threshold described in Section IV and V and the techniques described there. The NMI-values[2] of these clusterings lie in the interval $[0.647, 0.675]$[3] which is not bad. We combine these clusterings as described above and get a new clustering (see Fig. 6c) with an NMI-value of 0.706. This is better than any of the other clusterings and is on average an improvement of 0.046. This might not seem like a lot, but it solves two problems at once. If one uses the Levy Walk Clusterings algorithm there is a chance that the algorithm will give sub-par results. Without any experience in handling clustering algorithms one might be tempted in just using the first result and presenting it as the final solution. If one were to repeat the clusterings and combine them in this way, the chance that the result is sub-par becomes very small. This way it is even possible to deal with very bad clustering results. We have tempered with the thresholds to get some rather questionable results to prove this. When we now use four of the regular clusterings and four of those irregular clusterings and use the same technique as before, then the same result still hold up: We have NMI-values in the interval of $[0.462, 0.663]$[4] for the 8 clusterings. The combination of those (see Fig. 6d) give a clustering with an NMI-value of 0.667, which is again (slightly) better than any single one. We see that we can deal with sub-par clusterings an inexperienced user might get. This ensemble-approach is not very sophisticated, but already quite useful at this prototype-state.

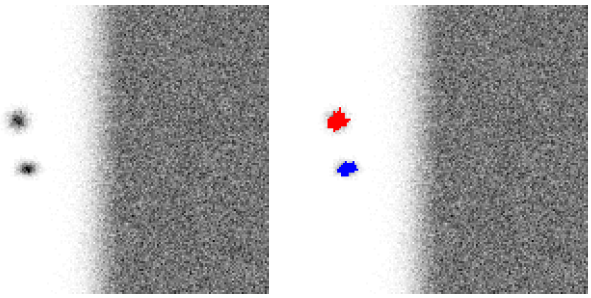
IX. Advantages of Levy Walk Clustering – artificial data

Sometimes it is easier to show the advantages of a clustering technique by comparing it to a strong opponent. For the type of datasets we are interested in here – many data points, with even more noise – one (if not the) choice for the clustering algorithm will be DBSCAN. Therefore, we have to show that we are better than it, at least in some cases. Now, besides the

[2]We use artificial data sets here and can therefore compute the NMI-value.

[3]The exact values (rounded to three decimal places) are: 0.647, 0.654, 0.663, 0.659, 0.665, 0.661, 0.673, 0.663.

[4]The exact values (rounded to three decimal places) are: 0.647, 0.654, 0.663, 0.659, 0.619, 0.596, 0.462, 0.580.



(a) The dataset: Romeo and Juliet against the world.

(b) The Cluster results – The clusters are found

Fig. 7. We can even find clusters if the noise is denser than the cluster.

fact that finding the parameter-combination for DBSCAN is of the utmost importance and often leads to quite false results, while Levy-clustering is (almost) parameter free, we also have one other quite important advantage, that we now want to emphasize. We take a look at the dataset depicted in Fig. 7. We have here two gaussian-shaped clusters and to their right a wall of noise. Following the golden rule that a cluster is supposed to entail the interesting part of the data, we want to find these clusters, but not the wall of noise.

If we were to use DBSCAN on this dataset, there are 3 outcomes possible: 1) No Cluster is found; 2) There are clusters found in the noise-area; 3) One or both of the clusters are found, but also one/many cluster(s) in the noise-area. This consideration holds true because the local density of data-points is higher in the noise-area, than in one of the clusters. The clusters have been constructed in a way that one of them is denser in its centre than the wall of noise, but not the other one. DBSCAN as a density-motivated clustering algorithm will now always find a "cluster" in the noise-area before it could find both of our clusters, simply because the local density there is far higher. Our Clustering algorithm on the other hand, has the more meaningful results. We clustered it 8 times to get a feel for the reliability of the results and the NMI-values for those did lie in the interval of $[0.720, 0.780]$. We used the technique described in Section VIII the 8 clusterings as input and got the clustering result depicted in Fig. 7b. The NMI-value for this combination is ≈ 0.768.

The Levy Walk Clustering algorithm here has the advantage, that it does not solely rely on the local density, but that it also considers the change in local density to be of importance. As the Levy Walk wanders in the direction of increased data points, it moves in the direction of the cluster maximum and invests its time there until it gets bored and wanders off. This behaviour allows us to swiftly explore the whole of the data set and find cluster despite a myriad of noise points, even if the noise is distributed most unfavourably.

Let us explore a somewhat more realistic setting and take a closer look at the dataset from the beginning with the nasty noise—Fig. 3a. We have here 5 gaussian shaped clusters of varying size and density, like they might be found in a real dataset. We did not find a data set with enough data points, that adheres to our needs and is also labelled – probably nobody wants to label a million data points – therefore we have to calculate the NMI-values for an artificial dataset. We will show that Levy Walk Clustering is very much capable of dealing with real data sets in Section X, but for NMI-Values we have to fall back on artificial data. We decided to use NMI-Values instead of AMI-Values, due to the easier implementation and when we compared AMI and NMI in external implementations the difference was minimal. The results can be seen in Table II.

TABLE II
NMI-VALUES FOR DATA SET 3A.

Algorithm	**NMI**
Levy Walk Clustering	0.561
Section VIII - Levy Walk Clustering	**0.7259**
DBSCAN	0.559
Sync	(0.05)
k-means	0.068

Since Levy Walk Clustering is non-deterministic, the NMI-Values listed in Table II can vary slightly. We created 24 clusterings and calculated the mean to get the NMI-Value for the Levy Walk Clustering. This way we get a somewhat significant value. The mean-deviation is around 0,03 though and hence the advantage against the DBSCAN Clustering – we tried the minPoints parameter in 10 points increments with eps in 0,01 steps until we got false clusters (eps=1,91 and minPoints=1120 seems to be a good choice) – is statistically insignificant. We can not say that the singular Levy Walk Clustering is better than the carefully parametrized DBSCAN, but if we use the technique from Section VIII the advantage becomes obvious and the lead is not in doubt. While DBSCAN can extrude the centre of the clusters, it can not find their tail. A parameter-combination that would add the tail too, would necessarily also add various false clusters. Levy Walk Clustering on the other hand is capable of ignoring such statistical flukes in noise distribution and focusing on the clusters. Due to its independence from absolute density values it is capable of guessing the correct clusters better. SYNC is in the current implementation not capable of handling such massive data sets and we need to use a thinned-out data set (see Figure 10) with only 2% of the data. On this data set SYNC found 11 clusters and got a rather low NMI-value.

X. TESTS ON REAL DATA

As we will see the current version of the Levy Walk Clustering algorithm is made for truly enormous data sets and such are hard to come by, especially when they are supposed to be classified to compute the NMI-/AMI-values. One data set we have found, that somewhat adheres to being labelled, is the North-Jutland-Street-data set [4]. While we do not have a real classification, we have a map of North-Jutland itself and can just look up which town is where and if we have found it. The dataset itself is far from trivial, so we used the technique described in Section VIII. We used 8 clusterings as

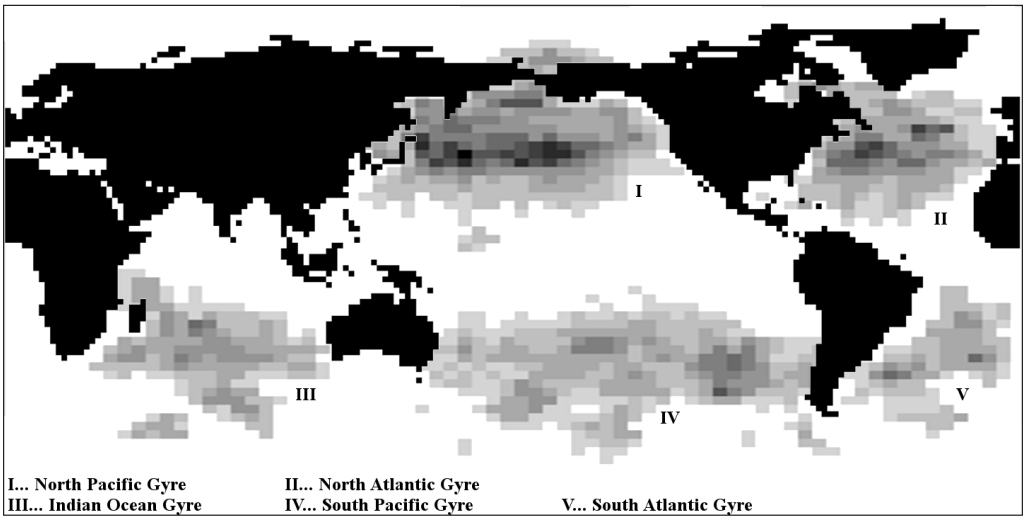


Fig. 8. A map of the world and the signal of the annual cycle that we found. We have a slight overlap with the continents because of the low resolution of the grid we used and the resulting tendency of the Levy Walk for imprecision. We tried to visualize the three-dimensional results by projecting it down to two dimensions, by plotting the two-dimensional layers of the found clusters. Overlapping layers are plotted darker. The tails of the found clusters (only one or two grid cells) are left out.

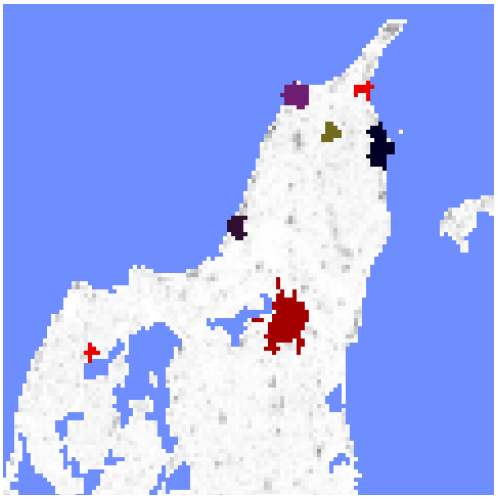


Fig. 9. Simulation results for the North Jutland Data set with $k = 120$—only the bigger towns are left

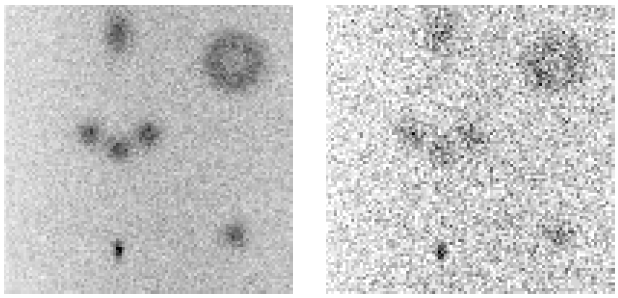
input with the proposed parameters. We went ahead with the first 8 results and the result is very satisfying.

The algorithm found the bigger settlements in North Jutland, while ignoring small ones, where only few people live. The biggest towns like Aalborg and Frederikshavn were easily found and put onto the map with their rough shape and their suburbs. Smaller villages like Nibe and Trend were also found. If one feels that there are too many small settlements in the clustering, one can reduce k. By doing this and repeating the clustering in the same way we get Fig. 9, which has the number of settlements reduced down to the biggest ones, with the advantage of improved runtime.

We have also taken a look at a more than two-dimensional dataset to show that we are not limited to them. We looked at the changes in climatological mean of ocean-temperatures from March to September[5]. The dataset already comes in the form of a grid, so we can use it directly instead of putting a grid over the data points. The three spatial dimensions make up the first three dimensions with the temperature-value as the fourth one. We have chosen the months March and September because due to the change in the zenith angle of the sun the change of temperature in this interval is the biggest. In March the water is usually at its coldest (in the northern hemisphere; the other way around for the southern one) and takes up energy until the ocean starts to cool again, when the sun is lower in the sky in autumn. Hence, the signal of the annual cycle should be clearest then. We scaled the grid from 360x180x20 down to 72x60x20, to improve on runtime and have only a few cluster as a result, which simplifies the analysis of the results. We got the following results: We almost always got the same clusters, each of them in a different ocean and corresponding to one of the big gyres present in an ocean. In the northern hemisphere are the North Pacific Gyre, which takes up most of the pacific ocean north of the equator, and the North Atlantic Gyre, the northward branch of which forms the Gulf Stream. In the southern hemisphere we found the South Pacific Gyre, the South Atlantic Gyre and the Indian Ocean Gyre. These 5 are all enormous currents in the oceans which are known for their influence on the global climate. The found clusters are also restricted to rather shallow depths and range from the surface to roughly 100m below sea level, indicating the depth to which the annual cycle of ocean temperatures penetrates. We summarize the results in Fig. 8.

All of these results are meaningful, though one might consider them trivial, if familiar with the subject. To corroborate them we asked Dr. Michael Mayer from the Department of

[5]NODC_WOA94 data provided by the NOAA/OAR/ESRL PSD, Boulder, Colorado, USA, from their Web site at http://www.esrl.noaa.gov/psd/



(a) A regular dense dataset (b) The same dataset with far lower density

Fig. 10. The same dataset with different densities.

Meteorology and Geophysics at the University of Vienna, to have a look a these results that we got and though we have to say that these results are not exactly the most fascinating ones for someone knowledgeable in this area of science, they – nevertheless – are meaningful. We can therefore say that our Clustering Algorithm delivers proper results here and is therefore not restricted to two dimensions.

XI. Limitations and Outlook

The size of the Grid G/The size of D: We have shown in Section VII the influence the size of the grid, determined by k, has. Fig. 5 seems to give the impression of an almost irrelevant choice of k concerning the NMI-value, as long as it is big enough. Very big k though will lead to a loss in the quality of clustering, expressed in a fall of the NMI-values. We take the same dataset as in Section VII again and cluster it with bigger values of k. If $k = 300$ the NMI falls below 0.6 and down to ≈ 0.3 for $k = 400$. This is because the direction the Levy Walk takes in a grid cell, becomes almost random, if the local density is too low. This becomes most obvious if we take a dataset and delete 90% of all data points (90% of the noise and 90% of the cluster points, see Fig. 10). In theory nothing much should have changed, but the Clustering Algorithm does not work any more. The direction the Levy Walk takes becomes too much dictated by chance and, of course, if the direction is random the whole concept of creating clusters breaks down. If we cluster the low-density-dataset the resulting NMI-value is not satisfying, but if we reduce k to a mere value of 30, we still get an NMI of ≈ 0.525, which is at least acceptable. We can conclude from all that, that the density – i.e. amount of data points per grid cell – has to lie above a certain threshold. If we take the dataset of Fig. 10 as representative, then we can conclude, that we should not go below 13-15 data points per grid cell. This in turn means that we need very big data sets for our clustering algorithm. If we have e.g. 5 dimension and a value of $k = 50$ (which is already rather small) we would need a data set of at least 4.687.500.000 data points for our algorithm to work.

Small data sets: As we have seen we are currently restricted to rather big data sets. We plan on abolishing this dependency for our algorithm by developing a re-sampling-technique that increases the data points. This way the Levy Walk should win its purposefulness back and should be capable of finding the essential information. This might also be possible by increasing the area the Levy Walk takes into account, when choosing a direction.

Getting rid of the Grid: We have a very clear idea on how we plan on getting rid of the grid, but due to restrictions in paper-length, we can not elaborate on that. We plan on publishing a non-grid version of the Levy Walk Clustering algorithm in the next step, where we will include more data sets.

XII. Conclusion

We have created a new approach to clustering using the hunting technique used by many predators, described by the Levy Walk-model. It is a clustering-algorithm which works on enormous data sets (though, sadly, currently only on those) and is almost parameter free. There are still parameters, like the size of the Levy Walk m, but none of them is difficult to be set. We have not conclusively proven how they should be set, but the simulations (Fig. 5) show good reason to believe, that we only need to surmount a certain threshold. The only parameter that we will most likely have to keep around is k, but this one is a choice of the wanted precision/runtime, not a necessity of clustering, like the parameters for DBSCAN. We showed that our algorithm is capable of handling massive data sets with enormous levels of nasty noise and that better than some well-known as well as state-of-the-art alternatives. While these comparisons were executed on artificial data, due to the lack of big enough data sets, we could show that it is very well capable of handling real datasets and creating meaningful results, which we were able to corroborate.

Future works now shall follow through with the removal of the grid we have used until now.

Acknowledgement

We would like very much to thank Dr. Michael Mayer from the Department of Meteorology and Geophysics at the University of Vienna with his help in analysing and corroborating the results for the climate-data in Section X.

References

[1] Böhm, C., Plant, C., Shao, J., Yang, Q., *Clustering by Synchronization*, KDD, 2010.

[2] Dempster, A. P., Laird, N. M., Rubin, D. B., *Maximum likelihood from incomplete data via the em algorithm*, Journal of the Royal Statistical Society, 1977.

[3] Dorigo, M., Stützle, T., *Ant Colony Optimization*, MIT Press, Cambridge, MA, 2004

[4] Kaul, M., Yang, B., Jensen, C.S., *Building Accurate 3D Spatial Networks to Enable Next Generation Intelligent Transportation Systems*, IEEE MDM, 2013.

[5] Monterey, G.I., Levitus, S., *Climatological cycle of mixed layer depth in the world ocean*, U.S. Gov. Printing Office, NOAA NESDIS, 1997

[6] Nakrani, S., Tovey, C., *On honey bees and dynamic server allocation in Internet hosting centers*, Adaptive Behavior, 2004.

[7] Yang, X.-S., Deb, S., *Cuckoo Search via Levy Flights*, NaBIC, 2009.

[8] Ester, M., Kriegel, H.-P., Sander, J., Xu, X., *A density-based algorithm for discovering clusters in large spatial databases with noise*, KDD, 1996.

[9] MacQueen, J. B., *Some methods for classification and analysis of multivariate observations*, Berkeley Symposium on Math. Stat. and Prob., 1967.