The COCO Conjecture and Its Applications in Clustering

Byron Gao and Robert Tung

1 Introduction

In this paper, we introduce a particular sequence of matrices that, for any set of n random variables, has the particular convergence behavior described in the next section. This sequence of matrices is related to the work of other researchers such as McQuitty and Clark, as well as Brieger, Boorman and Arabie, and the clustering, specifically bisection, algorithms that each set of researchers independently studied. This is due to the fact that the construction of these matrices and their resulting convergence, can be used as an algorithm that may be thought of as a generalization of those of said researchers, though we discovered the concept independently.

Clustering in particular is one of the most widely used techniques in data-mining, and is immensely important in its applications to various fields. Any application in which the identification of groups based upon shared behavior is significant may rely on the practice. Though many clustering algorithms have come about, and from a wide variety of approaches, there is no single generally-used clustering algorithm, as each will focus on different aspects of the data, and may optimize different factors. As such, this paper not only introduces our algorithm, but discusses its implications and uses as well.

This paper details the related concepts from relevant past papers, and introduces novel observations relating to the convergence behavior of this algorithm as well as many aspects of its use. This paper specifically addresses how to use this sequence of matrices as a means of clustering objects, and discusses the implications of our observations about the sequence/algorithm to the clustering application specifically. Finally, this paper discusses how some of these observations may be used to improve the efficiency of this algorithm, as one of the biggest issues with the similar algorithms in the past were the large time complexities and lack of scalability with large datasets.

2 Conjecture

Given n random variables, let M_1 be a symmetric $n \times n$ correlation matrix, where the (i, j)-th entry is the Pearson correlation coefficient between variable i and variable j. Based on a pre-determined set of indices, we select the corresponding columns from M_1 to form an $n \times r$ matrix M_1', where $2 \leq r \leq n$. We then compute the Pearson correlation coefficient for each pair of rows in M_1' to obtain a correlation matrix M_2. We continue with this process iteratively, producing a sequence of correlation matrices that results in one of the following three scenarios, with the second and third being extremely rare:

1. Common Convergence: The sequence converges to some correlation matrix M_∞, all of whose entries are either 1 or -1.

2. Uncommon Convergence: The sequences converges to some correlation matrix M_∞, whose entries are not all 1 or -1.

3. Undefined: The iteration terminates with a correlation matrix containing undefined values.

2.1 Efficiency Improvement

It can be noted that, as each matrix M_k is created specifically from M'_{k-1} rather than the whole matrix, then the other $n-r$ columns of M_{k-1} actually do not contribute at all to the calculation, and thus need not at all be calculated. However, this is actually true for all k, which can be seen by the following simple inductive argument. If we start with M_1 and form M'_1, we can definitely calculate all values in M'_2 from just M'_1, as M'_2 is nothing more than a submatrix of M_2. Significantly, the values in M'_2 are calculated with the same correlation process as would occur if we first calculated the entire M_2 and then isolated these columns. Thus, this is the same matrix. If we continue this process, we can see that the sequence of matrices not only has the same convergence behavior, but is composed of the same matrices, and thus converges in the same number of steps. Indeed, this was verified when run on over 10,000 randomly generated datasets. This is significant because it reduces the amount of calculations in each step, the most significant bottleneck in the time complexity of this algorithm according to Breiger, Boorman, and Arabie [1]. This brings our time complexity down to $O(r^3ni)$ for i iterations, rather than a complexity of $O(r^2n^2i)$ without this efficiency improvement, which can be a significant difference if a small or constant number of reference points are used. This time complexity comes from the fact that each correlation coefficient calculation is of complexity $O(r^2)$. Further efficiency improvements will be noted in a later section of this paper. Unless otherwise stated, other results and conjectures made in this paper can be assumed to take place under the original process of calculations for the sake of generality. However, unless they specifically involve entire matrices M_k instead of M'_k, they are indeed still valid under this more efficient process.

2.2 Further Generalization

Additionally, we have found that the using the cosine correlation coefficient instead of the Pearson correlation coefficient produces common convergence in the general case as well. This is significant in that it means our conjecture and our results may not be specific to the Pearson correlation coefficient, but instead may be applicable to any correlation coefficient satisfying certain properties. More work has yet to be done in this area, but likely the boundedness and the coefficient's -1 to 1 range are of significance. We also have yet to determine whether the other results in this paper are also applicable to the cosine correlation coefficient, and which results are applicable to the more general correlation coefficients that we are searching for.

3 Related Work

In their 1968 paper, McQuitty and Clark first explained what they described as "Iterative Intercolumnar Correlational Analysis" [2]. The method they described can be thought of as a subcase of ours, in which all entries of a row are used in each iteration of the matrix (the fact that they used columns instead of rows ultimately has no effect on the calculations as all matrices M_k are symmetric). In this study, they began by listing the benefits of such an approach, including that the approach could be used hierarchically to create a desired number of clusters, that it took account of all relationships in the set rather than just a subset such as the closest neighbors, that it implicitly created reasonable clusters, and that the algorithm would still function correctly under a broad factor of chance. To support their claims, they provided mathematical proofs to the claim that all objects of the same "type" (those that shared certain characteristics and should definitely be in the same cluster) ended up in the same cluster (due to 1's in the limiting matrix). They also proved that the algorithm intelligently put objects of opposite type in separate clusters. Next, they recreated these proofs with a level of chance factored in so as to show the generality of the argument.

Later, in their 1970 paper, McQuitty and Clark again investigated this problem and elaborated upon it [3]. They began by formally redefining a type for an object, which was broken down into more precise classifications. Acknowledging that the algorithm may at times create clusters of only one object, they distinguished between a group type and a single-object type. Perhaps more significantly, they realized that not all matrices resulted in what we in our study have come to call ”common convergence.” As such, they defined a system as a general set of objects that may share some characteristics, and a group quasitype as a set of objects for which some objects share some characteristics, but for which the entire set of objects do not necessarily share any characteristics. They continued by acknowledging some of the knife-edge cases (meaning that any slight shift would produce a matrix that was not of this case) that did not result in our notion of ”common convergence,” including the following, the implications and classifications of which we will discuss in the next section:

$$\begin{bmatrix} 1 & -\frac{1}{r-1} & -\frac{1}{r-1} & \cdots & -\frac{1}{r-1} \\ -\frac{1}{r-1} & 1 & -\frac{1}{r-1} & \cdots & -\frac{1}{r-1} \\ -\frac{1}{r-1} & -\frac{1}{r-1} & 1 & \cdots & -\frac{1}{r-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -\frac{1}{r-1} & -\frac{1}{r-1} & -\frac{1}{r-1} & \cdots & 1 \end{bmatrix} \text{and} \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix}$$

Finally, they noted in their paper that a proof of convergence for all but the exception cases had yet to be studied, and that there was a real need for this.

In 1973, Breiger, Boorman, and Arabie published a paper detailing a similar algorithm called CONCOR, in which they cited the previous two papers listed above [1]. Having independently discovered the algorithm (which we now consider the subcase of ours in which $r = n$) for the purpose of analyzing social networks, they also noted the ”knife-edge” characteristic of the cases that were not of ”common convergence,” and that none of these exception cases had ever come up in "more than one hundred applications to sets and subsets of network-related data ranging in size up to 70×70." Regardless, they did note one case that fits our notion of common convergence, but for which an intelligent clustering cannot be applied (which would result in convergence to a matrix of all 1's), listed below.:

$$M_0 = \begin{bmatrix} \frac{c_1 \times d_1}{n} & \frac{c_1 \times d_2}{n} & \frac{c_1 \times d_3}{n} & \dots & \frac{c_1 \times d_n}{n} \\ \frac{c_2 \times d_1}{n} & \frac{c_2 \times d_2}{n} & \frac{c_2 \times d_3}{n} & \dots & \frac{c_2 \times d_n}{n} \\ \frac{c_3 \times d_1}{n} & \frac{c_3 \times d_2}{n} & \frac{c_3 \times d_3}{n} & \dots & \frac{c_3 \times d_n}{n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{c_n \times d_1}{n} & \frac{c_n \times d_2}{n} & \frac{c_n \times d_3}{n} & \dots & \frac{c_n \times d_n}{n} \end{bmatrix}$$

They also noted the method of slightly altering values that we came across independently and that is described in section 3 of this paper.

In the matrix above, they specified that $\sum_i c_i = \sum_i d_i = n$, but, as we will later show, the sequence of matrices in this algorithm is invariant under linear transformations of the rows of the initial matrix. Thus, we provide the following generalization of this case, without the specifications on c_i and d_i provided above and with the generalization that any M'_k could be in this form. We do, however, provide the specification (which also must be satisfied for the above matrix) that no c_i or d_i is negative. If one were to be negative, the slope of the linear transformation between two rows would be negative and thus would result in a -1 in the following correlation matrix. For that case, this would in fact be a case of common convergence that is bisectable, as we will explain in a later section:

$$\begin{bmatrix} c_1 \times d_1 + x_1 & c_1 \times d_2 + x_2 & c_1 \times d_3 + x_3 & \dots & c_1 \times d_n + x_n \\ c_2 \times d_1 + x_1 & c_2 \times d_2 + x_2 & c_2 \times d_3 + x_3 & \dots & c_2 \times d_n + x_n \\ c_3 \times d_1 + x_1 & c_3 \times d_2 + x_2 & c_3 \times d_3 + x_3 & \dots & c_3 \times d_n + x_n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_r \times d_1 + x_1 & c_r \times d_2 + x_2 & c_r \times d_3 + x_3 & \dots & c_r \times d_n + x_n \end{bmatrix}$$

The reader will also note that this particular case is not a knife edge case, meaning a slight shift in values would indeed still produce the same clustering. However, any cases that are strictly not of common convergence, listed in later sections, do have this knife edge characteristic.

In addition, Breiger, Boorman, and Arabie's paper maintained that the speed of convergence was fairly rapid, with all of their examples reaching cutoff values of 0.999 or -0.999 within eleven iterations, often less. What was also significant was their observation that the algorithm could be applied on objects with multiple types of relationships, by stacking the initial matrices on top of each other to create a $kn \times n$ matrix where k is the number of types of relationships. Finally, they noted that the CONCOR algorithm was very close to optimizing within-group correlations, and was invariant under scalar addition. We have since shown that our algorithm, and thus the CONCOR algorithm, is also reasonably close to optimizing the k-means criteria, and is invariant under any linear transformation of its rows if the slope of said linear transformation is positive.

4 Discussion of Convergence Behavior

Let $a_1, a_2, ..., a_r$ be the set of indices of the columns of M_k of which M'_k consists. Now, consider the $r \times r$ matrix created by taking only the rows of indices $a_1, a_2, ..., a_r$ of M'_k, call this matrix M''_k.

4.1 Common Convergence

Note that for all matrices except for a few corner cases, this process will bring about common convergence. In fact, for all random data sets that we have generated, including those generated completely randomly or following some distribution, this has always been the case. The random datasets were generated with a set number of data points, using a random number generator built into Java with an upper bound that we chose, for 2-dimensional Euclidean data and the corresponding M_0 distance matrix. Those following a distribution were generated within those limitations (randomly for one dimension and set the other such that the points lied on a circle, setting all points in a grid, setting all points on a line, etc.) Finally, multiple toy datasets were used that were handpicked using the ”gen” tool available upon request.

Below is an example of common convergence.

Consider starting with the following matrix M_0, whose n rows represent n random variables.

$$\begin{bmatrix} 2.1 & 1.5 & 3.1 & 7.6 \\ 5.2 & 2.3 & 6.4 & 9.2 \\ 2.9 & 4.2 & 2.8 & 1.3 \\ 1.3 & 4.5 & 8.5 & 9.1 \end{bmatrix}$$

This results in the following $n \times n$ matrix M_1 below:

$$\begin{bmatrix} 1 & 0.909036663 & -0.924525321 & 0.703845711 \\ 0.909036663 & 1 & -0.988815153 & 0.640670958 \\ -0.924525321 & -0.988815153 & 1 & -0.554951004 \\ 0.703845711 & 0.640670958 & -0.554951004 & 1 \end{bmatrix}$$

Now consider the set of columns that we use to create M_1' to be the first, second and third columns, resulting in the following matrix M_1':

$$\begin{bmatrix} 1 & 0.909036663 & -0.924525321 \\ 0.909036663 & 1 & -0.988815153 \\ -0.924525321 & -0.988815153 & 1 \\ 0.703845711 & 0.640670958 & -0.554951004 \end{bmatrix}$$

Thus the matrix M_2 is as follows:

$$\begin{bmatrix} 1 & 0.9966059 & -0.997526718 & 0.999996481 \\ 0.9966059 & 1 & -0.999927188 & 0.996384007 \\ -0.997526718 & -0.999927188 & 1 & -0.997336742 \\ 0.999996481 & 0.996384007 & -0.997336742 & 1 \end{bmatrix}$$

This sequence of matrices eventually converges to the following matrix M_∞:

$$\begin{bmatrix} 1 & 1 & -1 & 1 \\ 1 & 1 & -1 & 1 \\ -1 & -1 & 1 & -1 \\ 1 & 1 & -1 & 1 \end{bmatrix}$$

Remember, however, that there is one case of common convergence that is unique from the rest, and that is when the entire matrix converges to a matrix of all 1's. This stems from some M'_k being in the following form:

$$\begin{bmatrix} c_1 \times d_1 + x_1 & c_1 \times d_2 + x_2 & c_1 \times d_3 + x_3 & \dots & c_1 \times d_n + x_n \\ c_2 \times d_1 + x_1 & c_2 \times d_2 + x_2 & c_2 \times d_3 + x_3 & \dots & c_2 \times d_n + x_n \\ c_3 \times d_1 + x_1 & c_3 \times d_2 + x_2 & c_3 \times d_3 + x_3 & \dots & c_3 \times d_n + x_n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_r \times d_1 + x_1 & c_r \times d_2 + x_2 & c_r \times d_3 + x_3 & \dots & c_r \times d_n + x_n \end{bmatrix}$$

Form 1

Note that this case implies an inability to bisect the set into two clusters as it converges to a matrix of all 1's (see next section). It can in fact be proven that this matrix can only be the form of some matrix M'_k with $k > 0$ if M''_k is, for some real x between -1 and 1, of the form given below (Note however that Form 1 can be the form of the initial random variables themselves, which would result in an nonbisectable common convergence):

$$\begin{bmatrix} 1 & x & 1 & x & \dots \\ x & 1 & x & 1 & \dots \\ 1 & x & 1 & x & \dots \\ x & 1 & x & 1 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

The proof of this is as follows: Assume for the sake of contradiction that some M'_k with $k > 0$ was in Form 1 above. Isolate M''_k from this matrix, which we recall is just a matrix composed of certain rows of M'_k. This matrix would have to be a square matrix with all 1's on the diagonal, but since it is in Form 1 above, all rows would have to be linear transformations of each other (possibly with negative slope, but not possibly with 0 slope). We will show this by inducting on the number of columns and rows of M''_k considered. First consider just two rows of M''_k, and the two columns of these rows whose indices match the row indices, call it $M''_{k\,2}$ (analogously to how we constructed M''_k). This 2×2 matrix would have to be of the following form, as the Pearson Correlation Coefficient is a commutative property:

$$\begin{bmatrix} 1 & x \\ x & 1 \end{bmatrix}$$

Now, for the sake of induction, assume that we take m rows and their respective columns of M''_k, call this $M''_{k\,m}$, and that it is of the form:

$$\begin{bmatrix} 1 & x & 1 & x & \dots \\ x & 1 & x & 1 & \dots \\ 1 & x & 1 & x & \dots \\ x & 1 & x & 1 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

If we then look at $M''_{k\,m+1}$, organized such that the first m columns of the first m rows form $M''_{k\,m}$, we see that the bottom-right corner has to be a 1 as this is a symmetric similarity matrix. Next, because linear transformations are one-to-one, the final row must be in the following form:

$$\begin{bmatrix} y & z & y & z & \dots & y & z & 1 \end{bmatrix}$$

Consider the case where $m+1$ is odd (an analogous proof would work if $m+1$ is even). From this we can determine that the entry to the left of the each y is an x and the entry to the left of each z is a 1. Thus, letting the linear transformation from the $m-th$ row to the $m+1-th$ row be expressed as $c*(M''_{k\,m+1})_m + d = (M''_{k\,m+1})_{m+1}$, we can then note that $c*x+d=y$ and $c*1+d=z$. Moreover, we know that the $m+1-th$ column of the matrix is the same as the $m+1-th$ row, so we also know from the right-most entries of the $m-th$ and $m+1-th$ rows that $c*z+d=1$. From $c*z+d=1$ and $c*1+d=z$ it can be gathered that $c=-1$ and $d=z+1$. Thus, $-x+z+1=y$. However, using a similar set of equations, it can be deduced that the linear transformation between the $m-1-th$ and $m-th$ row can be expressed as $-(M''_{k\,m+1})_{m-1}+x+1=(M''_{k\,m+1})_m$. Thus, using the right-most column of $M''_{k\,m+1}$, we can see that $-y+x+1=z$. Using this along with the equation $-x+z+1=y$, we can deduce that $z=x$ and $y=1$. Thus, the $m+1-th$ column and row are as expressed above. Therefore, by induction, M''_k as a whole must be of the form above.

However, the form above doesn't actually satisfy our restriction that all c_i and d_i be positive, as the slope between any two nonequal rows is -1, resulting in a bisectable limiting matrix. Thus, assuming the initial random variables are not of Form 1 then the case of nonbisectable common convergence is never reached.

4.2 Uncommon Convergence

At present it seems that cases of uncommon convergence arise when, for some k, the matrix M''_k is in one of the following forms (for any real number x between -1 and 1):

1. $$\begin{bmatrix} 1 & -\frac{1}{r-1} & -\frac{1}{r-1} & \dots & -\frac{1}{r-1} \\ -\frac{1}{r-1} & 1 & -\frac{1}{r-1} & \dots & -\frac{1}{r-1} \\ -\frac{1}{r-1} & -\frac{1}{r-1} & 1 & \dots & -\frac{1}{r-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -\frac{1}{r-1} & -\frac{1}{r-1} & -\frac{1}{r-1} & \dots & 1 \end{bmatrix}$$

2. $$\begin{bmatrix} 0 & x & x & \dots & x \\ x & 0 & x & \dots & x \\ x & x & 0 & \dots & x \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x & x & x & \dots & 0 \end{bmatrix}$$

3. $$\begin{bmatrix} 1 & x & x & \dots & x \\ x & 1 & x & \dots & x \\ x & x & 1 & \dots & x \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x & x & x & \dots & 1 \end{bmatrix}$$

4. Indices can be split into two subsets B=$\{b_1, b_2, ..., b_s\}$ and C=$\{c_1, c_2, ..., c_t\}$ so that if $i \in B$ and $j \in C$, $(M''_k)_{ij} = 0$, and such that the sum total of each of at least $n-1$ of the rows is 0. An example is given below, in which the subsets are $\{1, 3, 5\}$ and $\{2, 4\}$:
$$\begin{bmatrix} 1 & 0 & -0.75 & 0 & -0.25 \\ 0 & 1 & 0 & 1 & 0 \\ -1 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 \\ -0.5 & 0 & -0.5 & 0 & 1 \end{bmatrix}$$

It can be seen that if M''_k takes either the second or third form, then M''_{k+1} will be of the first form, and will stay in that form.

As for the fourth form, note that this is a precursor, and simultaneously a generalization, to the case that McQuitty and Clark brought up (the case for which the matrix contains "a set of identical off-diagonals with a set of +1's and -1's). With this matrix, the sequence of matrices would converge to one in which entries whose column index and row index are in the same subset are 1 or -1, and all other entries are 0, though which of the nonzero entries are 1's and which are -1's seems harder to predict at the moment.

In the future, it may be shown that these cases cannot arise as the correlation matrices of matrices that are not themselves in any of these cases. In particular, it may help to show that if row i and row j have the same correlation as row i and row l, then row j is a linear transformation of row l (perhaps under some restrictions based on dimension and data amount). If this were to be true, these convergence cases would be much more easily shown as unique. At present, it may still be the case that there are other cases of uncommon convergence.

It should also be noted that we can move from uncommon convergence to common convergence by shifting values minimally (e.g. by a factor of 0.000001). For example, if your M_1 matrix happened to be the following:

$$\begin{bmatrix} 1 & 0.5 & 0.5 & 0.5 \\ 0.5 & 1 & 0.5 & 0.5 \\ 0.5 & 0.5 & 1 & 0.5 \\ 0.5 & 0.5 & 0.5 & 1 \end{bmatrix}$$

then we could instead replace the matrix with:

$$\begin{bmatrix} 1 & 0.5 & 0.5 & 0.5 \\ 0.5000001 & 1 & 0.5 & 0.5 \\ 0.5 & 0.5 & 1 & 0.5 \\ 0.5 & 0.5 & 0.5 & 1 \end{bmatrix}$$

and as long as the shifted value is in one of the entries of that is used in creating M_1'', this lead certain entries to converge to 1's or -1's. However, it may be the case that more than one value needs to be changed, as the reader will note that the other random variables, whose index is not either the column or the row index of the changed value, are still symmetrically equivalent if only one similarity measure is changed. For example, for the following matrix:

$$\begin{bmatrix} 1 & 0.49 & 0.5 & 0.5 \\ 0.49 & 1 & 0.5 & 0.5 \\ 0.5 & 0.5 & 1 & 0.5 \\ 0.5 & 0.5 & 0.5 & 1 \end{bmatrix}$$

Will eventually converge to the following:

$$\begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -0.5 & -0.5 \\ 0 & 0 & -0.5 & 1 & -0.5 \\ 0 & 0 & -0.5 & -0.5 & 1 \end{bmatrix}$$

The reader will note that this is a different case of uncommon convergence, and not one of common convergence. However, shifting a second value whose column and row index are both neither 1 nor 2 for this particular example will result in common convergence. This shows us that multiple entries may have to be changed, but that each one provides a step toward common convergence. If the user were to use the clustering application listed later in this paper, it would be up to them to decide how to use this information based on what is best for their application (changing $\frac{r}{2}$ of the values or using the change method that we provide later in this paper). Moreover, it is significant that which value is changed and whether it is slightly increased or decreased will change the resulting matrix M_∞. This will be discussed further in the Applications section below.

4.3 Undefined Values

At present, it seems that the only cases in which undefined values arise are the following:

1. For some k, the matrix M'_k contains one row in which all entries are of equal value.
2. For some k, the rows of the matrix M''_k are all linear transformations of each other.

See below for examples:

1. M_k: $\begin{bmatrix} 0 & 5 & 5 \\ 5 & 0 & 5 \\ 5 & 5 & 0 \end{bmatrix}$, M'_k: $\begin{bmatrix} 5 & 5 \\ 0 & 5 \\ 5 & 0 \end{bmatrix}$, M_{k+1}: $\begin{bmatrix} undefined & undefined & undefined \\ undefined & 1 & -1 \\ undefined & -1 & 1 \end{bmatrix}$

2. M_k: $\begin{bmatrix} 1 & 6 & 2 & 1 \\ 7 & 4 & 5 & 6 \\ 3 & 8 & 10 & 12 \\ 5 & 12 & 15 & 18 \end{bmatrix}$, M'_k: $\begin{bmatrix} 6 & 2 & 1 \\ 4 & 5 & 6 \\ 8 & 10 & 12 \\ 12 & 15 & 18 \end{bmatrix}$, $M''_k.$: $\begin{bmatrix} 4 & 5 & 6 \\ 8 & 10 & 12 \\ 12 & 15 & 18 \end{bmatrix}$,

M_{k+1}: $\begin{bmatrix} 1 & -0.5 & -0.5 & -0.5 \\ -0.5 & 1 & 1 & 1 \\ -0.5 & 1 & 1 & 1 \\ -0.5 & 1 & 1 & 1 \end{bmatrix}$, M_{k+2}: $\begin{bmatrix} undefined & undefined & undefined & undefined \\ undefined & 1 & 1 & 1 \\ undefined & 1 & 1 & 1 \\ undefined & 1 & 1 & 1 \end{bmatrix}$

In the future, it may be shown that these cases cannot arise as the correlation matrices of matrices that are not themselves in either of these cases.

It can in fact be proven that, though there may be more cases that eventually lead to undefined values, every matrix with undefined values is immediately preceded by a matrix in the first form given above (meaning that every sequence of matrices leading to undefined values must have one matrix of the first form above). This is because of the observation that, representing one row being used to calculate the correlation coefficient as x_i's and the other as y_i's:

$$\sum_{i=1}^{n-2} {x_i}^2 \geq \frac{(\sum_{i=1}^{n-2} x_i)^2}{n}$$

This itself stems from the following simple algebraic proof:

$\forall i, j, {x_i}^2 + {x_j}^2 \geq 2x_i x_j$, so $\sum \frac{{x_i}^2 + {x_j}^2}{2} \geq \sum x_i x_j$, which, once dividing each side by $\frac{n}{2}$, simplifies to

$$\sum_{i=1}^{n-2} {x_i}^2 \geq \frac{(\sum_{i=1}^{n-2} x_i)^2}{n}$$

What this means is that each of the two expressions in the square-root of the denominator of the Pearson product moment correlation coefficient equation is nonnegative. Because this stems from each x_i and x_j satisfying:

$${x_i}^2 + {x_j}^2 \geq 2x_i x_j$$

we know that equality to 0 of each expression in the denominator of the correlation coefficient equation, and thus equality of $\sum_{i=1}^{n-2} {x_i}^2$ to $\frac{(\sum_{i=1}^{n-2} x_i)^2}{n}$, can only hold when all the x_i's (or equivalently all the y_i's) are the same.

It may, however, be the case, that there are more precursors to the first form that we have not noted here.

In a similar manner to the case of uncommon convergence, we can shift slightly the values in whatever M_k' led to these corner cases. For this case, however, since the undefined values are a result of one row of M_k'' being of all equal values, we definitively need only shift one value in each such row to avoid the undefined values (unlike the case of uncommon convergence, in which many values may need to be shifted in order to bring us to common convergence). However, this does not mean that these shifts alone will lead us to a case of common convergence, as it may be a case of uncommon convergence.

For example, begin with the M_1' matrix from the first case, given again below:

$$\begin{bmatrix} 5 & 5 \\ 0 & 5 \\ 5 & 0 \end{bmatrix}$$

Let us shift one value slightly, as below:

$$\begin{bmatrix} 5.000001 & 5 \\ 0 & 5 \\ 5 & 0 \end{bmatrix}$$

This will lead to the following M_2:

$$\begin{bmatrix} 1 & -1 & 1 \\ -1 & 1 & -1 \\ 1 & -1 & 1 \end{bmatrix}$$

Again, the matrix M_∞ will vary based on what value is changed and whether it is increased or decreased, as will be discussed below.

5 Application

These observations on the convergences of correlation matrices become incredibly important when considered in the context of data clustering, specifically the bisection of datapoints. Consider a set of n objects, and compute M_0, in which the i,j-th entry is the distance between object i and object j. Note that any objects that have some form of distance measure can be compared in this way. Now, create M_0' and M_1 in an analogous way to the previous scenario, and create all subsequent matrices in the same way as well. Call the columns that we use to create each M_k' the ”representative columns” and similarly call the objects with those indices ”representative points”.

For all datasets that lead to common convergence, we can note that the matrix M_∞ is not just a matrix of all 1's and -1's but in fact is specifically a matrix such that if we permute the columns in some manner, and then permute the rows in the same manner, we can create a matrix that is in the following form, (Note that this matrix represents the same information as before permuting the columns and rows, but with the indices effectively renumbered):

$$\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$$

where each 1 represents a matrix of all 1's and each -1 represents a matrix of all -1's. In the context of our application to bisection, those objects whose indices make up the columns and rows of the top-left submatrix are put in one cluster, and similarly for the bottom-right submatrix. This is just as was noted by Brieger, Boorman and Arabie for the CONCOR algorithm [1], though we have now extended the conjecture to our algorithm.

5.1 Rough Explanation of Clustering

Since this algorithm is not formulated as an optimization problem, it is worth noting what the algorithm actually does. One of the best ways to convey this is to explain why two points that are very close may not, in fact, end up in the same cluster, while two points that are very far away may, in fact, end up in the same cluster. Consider the following dataset:

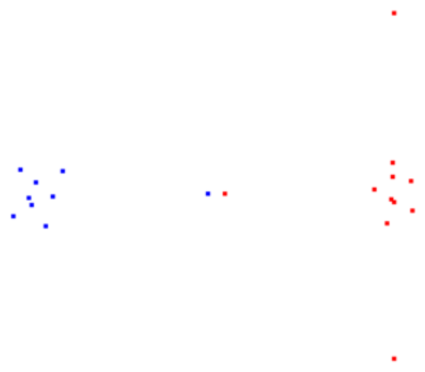

Figure 1: Example of far and close points

Note that the two points in the center are very close, but are in fact in different clusters. It is indeed true that the entry in M_1 whose column and row indices make up the indices of these two points is actually very close to 1. However, the corresponding entry in M_∞ is -1. When we consider the calculation of the Pearson Product Moment correlation coefficient, we can visualize the data as in Figure 2 below, in which the horizontal axis is the distance from object i to object l for each point l and the vertical axis is distance from object j to object l for each point l when we are correlating rows i and j.

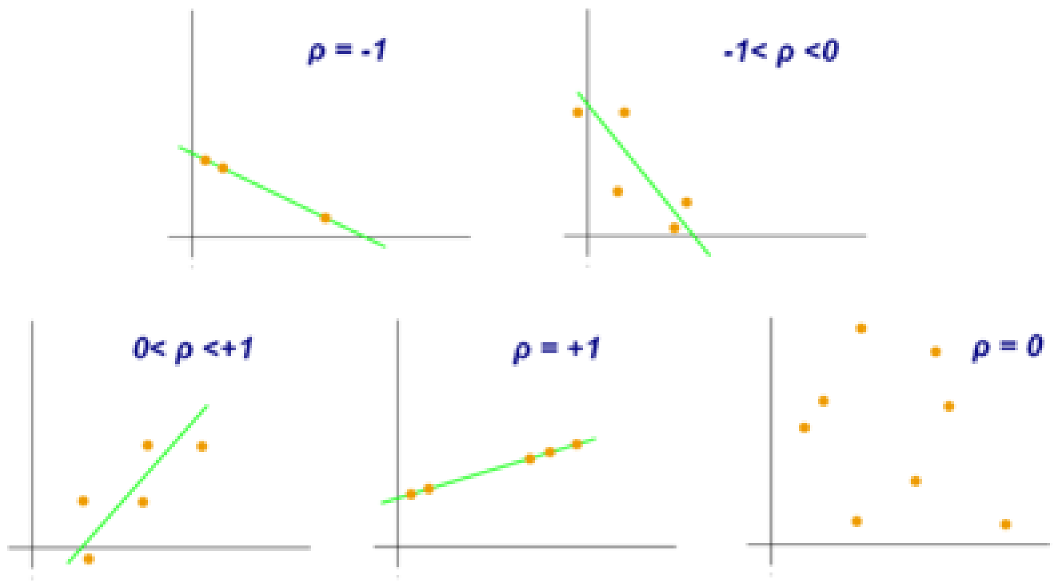

Figure 2: Correlation Coefficient Visualization, from Wikipedia

In the general algorithm (when not applied specifically to the clustering of objects), each axis is one of the two random variables and each point represents a point whose two dimensions are the random variables we are correlating.

The reason that two close points may end up in different clusters stems from the fact that the correlation coefficient calculation is influenced by whether the higher values are higher for both variables and lower values are lower for both variables (e.g. if the lowest x-axis value is from the same point as the lowest y-axis value, this pushed the correlation coefficient up toward

1). Thus, though two points may be very close, they may relate to other points differently. In the example above, the left point is closer to all points to the left of the two center points than those to the right, and the right point has the opposite behavior. Though their closeness may outweigh this in the first matrix, as the distance values of M_0 are very similar, the correlation between values in the left cluster and values in the right cluster will only become more and more distinct, making the fact that each point is closer to one side increasingly important in the resulting calculations. With the same reasoning, it can be seen that the point far above the right cluster and the point far below the right cluster are in the same cluster due to the fact that they are closest to largely the same points and furthest from largely the same points, and that they have similar similarity measures to all points.

5.2 Nonbisectable Common Convergence

Remember, however, the specific case that results in a matrix of all 1's (effectively a matrix of the same block form as above but with size 0 blocks in the top-right, bottom-left and bottom-right). Remember that this occurs when some M'_k is of the following form, where all pairs $c_i \times d_j$ form a number of the same sign (all positive or all negative):

$$\begin{bmatrix} c_1 \times d_1 + x_1 & c_1 \times d_2 + x_2 & c_1 \times d_3 + x_3 & \dots & c_1 \times d_n + x_n \\ c_2 \times d_1 + x_1 & c_2 \times d_2 + x_2 & c_2 \times d_3 + x_3 & \dots & c_2 \times d_n + x_n \\ c_3 \times d_1 + x_1 & c_3 \times d_2 + x_2 & c_3 \times d_3 + x_3 & \dots & c_3 \times d_n + x_n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_r \times d_1 + x_1 & c_r \times d_2 + x_2 & c_r \times d_3 + x_3 & \dots & c_r \times d_n + x_n \end{bmatrix}$$

Form 1

We noted before that for $k > 0$, this would mean that M''_k would have to be of the following form, and that it was in fact unattainable:

$$\begin{bmatrix} 1 & x & 1 & x & \dots \\ x & 1 & x & 1 & \dots \\ 1 & x & 1 & x & \dots \\ x & 1 & x & 1 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

Form 2

However, it can be shown, through an analogous proof to the previous, that for this application, since we use a distance matrix to begin with instead of directly calculating the correlations between random variables, if it were the M_0'' matrix that was in Form 1, then it would in fact have to be of the following form, based on the requirement of 0's along the main diagonal and a similar proof to the previous:

$$\begin{bmatrix} 0 & x & 0 & x & \dots \\ x & 0 & x & 0 & \dots \\ 0 & x & 0 & x & \dots \\ x & 0 & x & 0 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

Form 3

We have already shown that for any initial random variables, which can be expressed as some M_0, Form 2 is unattainable and thus this applies to this particular application as well. Now we note also that if M_0'' is of Form 3, then by default all odd-indexed objects would be equivalent and all even-indexed objects would be equivalent, as there is a distance measure of 0 between them. Thus, for any set of distinct objects, nonbisectable common convergence cannot arise under this algorithm.

5.3 Convergence Proof: $r = 2$

For the case of $r = 2$, it can be proven that the sequence of matrices converges to a matrix under common convergence as long as no point is equidistant from both reference points (if a point is in fact equidistant from both reference points, this will lead to undefined values). This is due to the fact that, for any pair of objects i and j that are each not equidistant from both reference points, one of the following occurs:

1. The first object is farther from the first reference point than the second reference point, and the same goes for the second object.

2. The first object is closer to the first reference point than the second reference point, and the same goes for the second object.

3. The first object is farther from the first reference point than the second reference point, and the second object is closer to the first reference point than the second reference point.

4. The first object is closer to the first reference point than the second reference point, and the second object is farther from the first reference point than the second reference point.

With a bit of algebra, or by visualizing the Pearson Correlation Coefficients of the rows of M'_0 on a 2-D Euclidean graph, it can be seen that for the first two cases, $(M_1)_{ij} = 1$, and for the last two cases, $(M_1)_{ij} = -1$. This is illustrated in Figures 3 and 4 below.

We will show in a later section that if $(M_k)_{ij} = 1$, then $(M_p)_{ij} = 1$ for all $p > k$ as well (and the same for -1), so this M_1 is also the matrix M_∞ that this sequence converges to. Note that this reasoning also applies when object i or object j is in fact a reference point, as they are by default closer to themselves and farther from each other.

If, in fact, one object, call it object i, is equidistant from both reference points, then row i of M'_0 is a multiple of the vector (1,1), so it falls into one of the cases of undefined values we

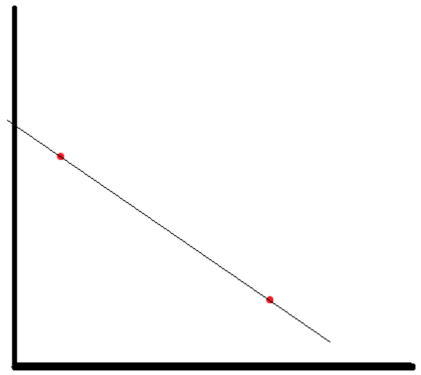

Figure 3: Points closer to different references

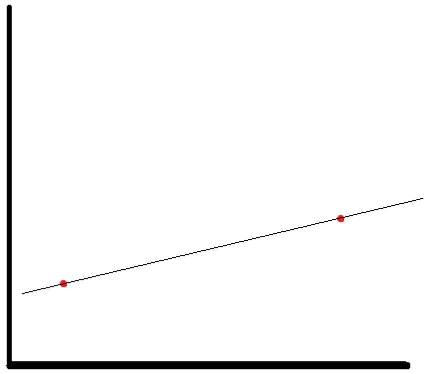

Figure 4: Both points closer to same reference

noted previously, and thus does not lead to a convergence.

5.4 Convergence Edge Cases

Note that, for any r, the first case of undefined values in our matrix observations can be translated to one in which an object is equal in similarity/equidistant from all representative points, and thus it cannot be determined which cluster it should be a part of. In this way, the adjustment we described in section two, in which one value is shifted marginally, has the following effect. If we shift a value of M_0 up, then the object whose index matches the row of the shifted value will be shifted away from the representative point whose index represents the column of the shifted value. Thus, this point will be moved to the opposite cluster from that representative point. If instead the value in M_0 is shifted down slightly, the object will be moved to the same cluster as that representative point. Note that either move is justified as the point is equidistant from both, and that both will result in common convergence, and note also that changing values in M_k for $k > 0$ has the opposite effects to those described above.

Note that the last case of uncommon convergence can arise when one reference is equidistant to all other references and those other references are all symmetrically equivalent, or just when all reference points are symmetrically equivalent. Examples of both of these cases are given below:

Figure 5: Uncommon Convergence Case 4: Extra Point

Figure 6: Uncommon Convergence Case 4: All Symmetric

When the correlation matrices are calculated for these cases, the only nonzero entries are 1's when the column index equals the row index and -1's when the two indices represent points that are opposite corners of the square. This can be interpreted as the fact that those points, whose distance is the farthest in the graph, are definitely in separate clusters, but all other points are pulled evenly toward each of these clusters. Thus, a value of 0 (or actually a value converging toward zero but never reaching it in a finite number of iterations) is left in the other entries. There may, however, be other cases that still satisfy this case of uncommon convergence.

Finally, note that the first three cases of uncommon convergence can be translated into one in which the representative points are all equidistant/of-equal-similarity to each other (and thus symmetrically equivalent in this way), and thus they cannot be clustered into two groups, leading to an inability to bisect the data set as a whole. Note also that for all of the first three cases noted in the previous section, the sequence of M''_k matrices will converge to the first case of uncommon convergence:

$$\begin{bmatrix} 1 & -\frac{1}{r-1} & -\frac{1}{r-1} & \cdots & -\frac{1}{r-1} \\ -\frac{1}{r-1} & 1 & -\frac{1}{r-1} & \cdots & -\frac{1}{r-1} \\ -\frac{1}{r-1} & -\frac{1}{r-1} & 1 & \cdots & -\frac{1}{r-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -\frac{1}{r-1} & -\frac{1}{r-1} & -\frac{1}{r-1} & \cdots & 1 \end{bmatrix}$$

Similar to the case of undefined values, the adjustment process we described above is justified in any case of uncommon convergence. If we slightly increase one value in the M_0 matrix, this would push the two representative points, whose indices are the column and row indices of the shifted value, farther apart, leading to placement in opposing clusters. A slight decrease in the value in M_0 would instead result in them being put in the same cluster. Note also that this shift breaks the symmetry of the points and, as at least one asymmetric point is being used as a reference this brings us away from this corner case and into one of common convergence.

We propose the following way to solve this issue for the case of uncommon convergence, in a general manner that doesn't require changing the data itself. This would be to add a value of $\frac{(-1)^{ij+I}\times(0.00001)}{(i+j)^I}$ where I is the iteration number of the matrix and i and j are the row and column index respectively. Because this is such a minimal change in the data, the clustering should not be adversely affected. Moreover, because the values in each row and column are being shifted in different directions, this will lead us away from one of the cases of uncommon convergence in M_{k+1} should one occur in M_k. The reason that the factor of I is also necessary is so that if we happen to move from a case of common convergence to a case of uncommon convergence in one iteration, then it would still be changed in the next; it also helps us in that the values do not move significantly from what they would be without this correction, regardless of how many iterations are used in the calculation. Moreover, the reason we need to divide by the denominator given above is to account for the fact that more than one value may need to be changed to reach common convergence (as detailed in the previous section), and in this way, all values in the matrix are changing in different ways with each iteration, so no matter what values we are observing in the matrix, we will never have a set of the variables stay symmetrically equivalent.

Note, however, that because the case of undefined values is not one in which a matrix in one of the forms listed previously can be remedied in the next iteration (e.g. if our general remedy happend to lead M_k to have a row of all the same value, the next iteration would have undefined values, which cannot be changed back to the original values by some general adjustment of values). As an example, note that adding 0.01 to the 1,2-th entry of

$$M_k'' = \begin{bmatrix} 0.5 & 0.5 & 0.5 \\ 0.5 & 1 & 0.5 \\ 0.5 & 0.5 & 1 \end{bmatrix}$$

would steer this away from undefined values, but it wouldn't be a good general fix as it would lead to undefined values in this matrix:

$$M_k'' = \begin{bmatrix} 0.5 & 0.49 & 0.5 \\ 0.5 & 1 & 0.5 \\ 0.5 & 0.5 & 1 \end{bmatrix}$$

The reason that the cases of uncommon convergence could be adjusted as such was because, prior to this remedy, M_k and M_{k+1} were equivalent under the cases of uncommon convergence, and thus the remedy could be applied to either iteration. But because the case of undefined values needs only 1 iteration to terminate, these cases can only in general be remedied by either running the algorithm once, noting if undefined values arise (and at what iteration they would arise), and correcting the values on a second run of the algorithm, or by checking at each step whether there are any rows in which all values in reference columns are the same. That is not to say that we cannot take a specific application and adjust the data to stray from undefined values, but merely that the same adjustment would not work for every dataset.

5.5 Distance Measures with Triangle Inequality

For any objects whose distance measure follows the triangle inequality, we also propose the hypothesis that we need only check M_0 (and the resulting M_0' and M_0'') for rows whose entries are all equal (and thus undefined values somewhere in the sequence), rather than checking for these for all M_k. This is because rows whose entries are all equal would imply equal correlation from the object whose index is that of the row to all representative objects, which we hypothesize cannot occur unless the object in question is of equal distance to all representative objects.

6 Observations and Resulting Minor Efficiency Improvements

Using the formula for Pearson Correlation Coefficients, it was first shown that each matrix in our algorithm, and thus the resulting bisection as a whole, was invariant under linear transformation of rows, as long as the slope of the linear transformation is greater than zero (note that with an analogous proof, it can be shown that if the slope is less than zero, the Pearson Correlation Coefficient is exactly negative of what it was before linear transformation). The algebraic justification for this is stated below:

Consider the Pearson Correlation Coefficient between two vectors $(x_1, x_2, ...x_m)$ and $(y_1, y_2, ..., y_m)$ and compare this to the Pearson Correlation Coefficient between $(a \times x_1 + b, a \times x_2 + b, ..., a \times x_m + b)$ and $(y_1, y_2, ..., y_m)$ for some constants a and b. The second Pearson Correlation Coefficient can be given by the following formula:

$$\begin{aligned}
r &= \frac{\sum_{i=1}^{m}(a\times x_i+b)\times y_i - \frac{\sum_{i=1}^{m}(a\times x_i+b)\times\sum_{i=1}^{m}y_i}{m}}{\sqrt{(\sum_{i=1}^{m}(a\times x_i+b)^2-\frac{(\sum_{i=1}^{m}(a\times x_i+b))^2}{m})(\sum_{i=1}^{m}{y_i}^2-\frac{\sum_{i=1}^{m}{y_i}^2}{m})}}\\
&= \frac{a\sum_{i=1}^{m}x_iy_i+b\sum_{i=1}^{m}y_i-a\frac{\sum_{i=1}^{m}x_i\sum_{i=1}^{m}y_i}{m}-\frac{mb\sum_{i=1}^{m}y_i}{m}}{\sqrt{(a^2\sum_{i=1}^{m}{x_i}^2+2ab\sum_{i=1}^{m}x_i+mb^2-\frac{a^2(\sum_{i=1}^{m}x_i)^2}{m}-\frac{2amb\sum_{i=1}^{m}x_i}{m}-\frac{m^2b^2}{m})(\sum_{i=1}^{m}{y_i}^2-\frac{\sum_{i=1}^{m}{y_i}^2}{m})}}\\
&= \frac{a\sum_{i=1}^{m}x_iy_i-a\frac{\sum_{i=1}^{m}x_i\sum_{i=1}^{m}y_i}{m}}{\sqrt{(a^2\sum_{i=1}^{m}{x_i}^2-\frac{a^2(\sum_{i=1}^{m}x_i)^2}{m})(\sum_{i=1}^{m}{y_i}^2-\frac{\sum_{i=1}^{m}{y_i}^2}{m})}}\\
&= \frac{a(\sum_{i=1}^{m}x_iy_i-\frac{\sum_{i=1}^{m}x_i\sum_{i=1}^{m}y_i}{m})}{a\sqrt{(\sum_{i=1}^{m}{x_i}^2-\frac{(\sum_{i=1}^{m}x_i)^2}{m})(\sum_{i=1}^{m}{y_i}^2-\frac{\sum_{i=1}^{m}{y_i}^2}{m})}}\\
&= \frac{(\sum_{i=1}^{m}x_iy_i-\frac{\sum_{i=1}^{m}x_i\sum_{i=1}^{m}y_i}{m})}{\sqrt{(\sum_{i=1}^{m}{x_i}^2-\frac{(\sum_{i=1}^{m}x_i)^2}{m})(\sum_{i=1}^{m}{y_i}^2-\frac{\sum_{i=1}^{m}{y_i}^2}{m})}}
\end{aligned}$$

Note that the last line is precisely the Pearson Correlation Coefficient formula for $(x_1, x_2, ...x_m)$ and $(y_1, y_2, ..., y_m)$. What this means is that if we perform a linear transformation with positive slope on one of the random variables, then the M_1 matrix produced would be exactly the same, leading to the same M_∞ matrix by default.

What this implies is that one can normalize the starting matrix under linear transformation, in order to provide an easier comparison between datasets, and still be confident that the same results are produced. Breiger, Boorman and Arabie noted a small part of this when they acknowledged the ability of their algorithm, a subcase of ours, to be invariant under the addition of a scalar to all values in the matrix. They also noted that it was not invariant under general monotonic transformation [1].

We also showed algebraically that if $(M_k)_{ij} = 1$, then $(M_{k+1})_{ij} = 1$, and similarly for -1. The algebraic justification for this is stated below:

If $(M_k)_{ij} = 1$, then the following is true for the rows of M_{k-1}: $(M_{k-1})_i = a(M_{k-1})_j + b$ for some constants a and b with $a > 0$. Thus, since we have proven that the Pearson Correlation between row j of M_{k-1}, noted as $(M_{k-1})_j$, and any vector is the same as the Pearson Correlation between $a(M_{k-1})_j + b$ and any vector, this implies that the Pearson Correlation Coefficient between $(M_{k-1})_j$ and any $(M_{k-1})_l$ is the same as the Pearson Correlation Coefficient between $(M_{k-1})_i$ and any $(M_{k-1})_l$. Thus, for all $1 \leq l \leq n, l \neq i$ and $l \neq j$, $(M_k)_{li} = (M_k)_{lj}$. But $(M_k)_{ii} = (M_k)_{ij} = (M_k)_{jj} = 1$ also holds, so $(M_k)_i = (M_k)_j$ and thus $(M_{k+1})_{ij} = 1$. Therefore, inductively, this is true for all following matrices. An analogous proof can be given for $(M_k)_{ij} = -1$, in which $a < 0$.

This means that as soon as we see for some k, i, and j that $(M_k)_{ij} = 1$, we know that object i and object j will be in the same cluster, and as soon as we see for some k, i, and j that $(M_k)_{ij} = -1$, we know that object i and object j will be in opposite clusters. This leads to the following efficiency improvement: For every 1 or -1 we find as $(M_k)_{ij}$ (or for every number past a certain threshold e.g. above 0.999 or below -0.999), we can remove either row i or row j and its respective column. In its place, we can keep track of which objects are definitely in same and definitely in different clusters. We can then put that information together at the end, saving us many calculations of correlation coefficients, which are each of O(r^2) time. However, note that, if the object is one whose index represents a reference column, the removal of the column with this index leads to a loss of accuracy, as we now lose the comparison of other objects to the two whose indices are relevant here. However, there is no way to remove the row without removing the column in future iterations M_k, so the user may still want to make this efficiency improvement. For objects that are not reference objects, this is not a problem, and the row and column can be removed without worry. The user may also want to devise a method of choosing new reference columns should the current reference columns be removed, though we will not discuss such methods here, as the correct approach may largely depend on the context in which it is used. The actual method for doing performing the efficiency improvement listed above can be performed as stated below:

We can create a linked list of hashsets, each representing a group of objects that we know to be in the same cluster. After each iteration of calculating the new M_k, we can check for 1's and -1's (or numbers that are sufficiently close to be considered 1's and -1's). For each 1 we find (aside from those along the main diagonal), we can keep track of the fact that the column index and row index of that entry are indices of two objects that are certainly in the same cluster. Check if the first, call it object x, is in any of the sets in the linked list. If it is, add the other, call it object y, to that set. If that object y was already in another set in the linked list, merge every element in the latter set with the former, and clear the latter set from the overall linked list. If x was not in any of the sets, do the analogous procedure with y, adding x and possibly its group if relevant. If after all these checks, neither element has been added, then hash those two values and append the set containing both to the end of the linked list. Once finished doing so, remove column x and row x from any further calculations in order to increase efficiency (Noting the sacrifice of accuracy as the relationship between other objects to object x is removed). In each iteration of the matrix, track also any -1's found, and the fact that the two indices involved represent objects in separate groups. This can be stored as a linked list of tuples. Then remove the row and column of the first index as before. Once the entire matrix has converged to 1's and -1's, all of our information is stored in the linked list of hashsets used to describe things in the same cluster, and this list of tuples used to describe things in separate clusters. It is easy to see that the list of tuples representing objects in different clusters, can be used to provide the information needed to combine the hashsets of the first linked list as needed, thus providing the final clustering. The problem above is analogous to one of ”dynamic connectivity” for which there are many solutions, some of which optimize the speed of unionizing the sets, and some of which optimize the speed of finding out whether two objects are in the same set. For this application, since the ”find” procedure is unnecessary until the very end, we can use any algorithm that optimizing the unionizing process.

It may also be considered to sacrifice accuracy for efficiency by having the same method of early object removal, but for values which have been positive and monotonically increasing or negative and monotonically decreasing for a predetermined number of iterations. With a large enough number of required iterations, accuracy can be achieved, but must be balanced with the diminishing amount of efficiency gained from larger amounts of iterations required. The actual process of this improvement can be done by constructing a second ”state” matrix with the same dimensions as M_k, but instead having a 0 if the value just changed signs, an integer $m_+ > 0$ if the value has been positive and monotonically increasing for m_+ iterations, and an integer $m_- < 0$ if the value has been negative and monotonically decreasing for $-m_-$ iterations.

Finally, a small efficiency improvement can be made by only calculating the values strictly above the main diagonal of the matrix M_{k+1} from the values in M_k, as the calculation of the Pearson Product Moment correlation coefficient is a symmetric operation. Thus, after calculating those values above the diagonal, we can fill in those below the diagonal accordingly. This can be done even when only some of the points are reference points, as we can just set both the entry below the diagonal and the entry above the diagonal to the correlation value found and we can add a conditional (which is of constant time) to check each time whether the value has been calculated previously. Moreover, the Pearson Product Moment Correlation Coefficient is always 1 for two identical vectors, and thus the main diagonal is always populated by 1's, so these values can be prepopulated every time.

Perhaps more significantly, note that because our algorithm can be performed for any amount of reference points, and with any initial set of random variables, we can actually switch the set of reference points in the middle of the algorithm, as long as we make only a finite amount of changes in reference points. For example, we may use five reference points in the first three correlation calculations, then consider the resulting matrix to itself be a starting set of random variables, and continue the correlation calculations with a different set of reference points, perhaps three or four. This provides not only a way to avoid certain cases that aren't of common convergence (such as the case where all reference points are equidistant), but it also creates an efficiency improvement as we can choose less reference points for some steps in the process. In terms of the convergence behavior, uncommon convergence, due to its being defined as all M_k after a certain point being a certain matrix, would only occur if the final reference set leads to uncommon convergence, analagous (but not necessarily equivalent in terms of matrix values) to the case if we had just started with this reference set through all calculations. It should be noted however, that if any reference set causes undefined values at some point, then the process would terminate at that edge case, and thus that is an edge case that is not specific to only the final reference set. These cases are, however, still knife-edge cases, and thus a general, randomly generated matrix should, with all probability, create a case of common convergence. However, as there are certainly some sets of reference points that are more intelligent than others, the user of this algorithm should be careful not to sacrifice extensive accuracy for this level of efficiency.

It is also incredibly important to note that, if we change the set of reference points infinitely, it still converges for the general case, and any exceptions to this have a similar knife-edge characteristic relating to either the structure of the dataset or the way in which reference points were changed. In fact, we ran the algorithm on over 25,000 randomly generated datasets with reference points being randomly selected from a subset of these each time, and every time the matrix resulted in common convergence. However, there are absolutely still knife-edge cases that do not converge. Take for example, the points in the 3-dimensional Euclidean Plane (0,0,1), (0,1,0), (1,0,0), (0,0,2), (0,2,0), (2,0,0) as the dataset, and oscillate the reference set between being the first three points and being the last three points. For this case, due to the fact that the reference set is always symmetrically equivalent, the matrix will not converge to common convergence.

Breiger, Boorman and Arabie also noted that the CONCOR algorithm was very close in performance to an algorithm that maximized mean within-group correlations. With our algorithm, we showed empirically that, if all or almost all points were used as references, our algorithm was very close (and for some datasets and some choices of references it was exactly the same) in performance to an algorithm that maximizes mean within-group correlations. For these tests, simulations were run on sets of randomly generated data points within a certain range for 2-dimensional Euclidean data, as well as for many hand-selected toy data sets of 2-dimensional data. The mean within-group correlations were found by averaging the correlation values in the M_1 matrix for those entries whose index and column indices represented objects that ended up in the same cluster for mean within-group correlations relative to the group. In this vein, we also took the time to test whether using all points as reference points provided the highest mean within-group correlation values when compared to using other subsets of the points as reference sets, and when compared to all possible clustering combinations. For datasets of 10, 15 and 20 points of 2-dimensional Euclidean data, we found that using all reference points was almost always tied for the best mean within-group correlation (rarely not the best and rarely the only reference set that created the optimal clustering for mean within-group correlations). All the other reference sets that produced same or better mean within-group correlations seemed to use a majority of the points anyway. This is significant in that if it were not the case, we may be able to devise a way to select a constant number of reference points, r, to produce an intelligent clustering, which would increase the efficiency and thus the scalability of this algorithm significantly. However, it seems that this is not the case. We hypothesize that the reason all reference points is not strictly always the best is that the algorithm does not optimize the values in the M_1 matrix, but is rather closer to optimizing those in the M_2 matrix, even closer to optimizing the values in the M_3 matrix and so on. This is worth looking into more in the future.

Similarly, we showed that, with enough reference points, our algorithm is close in performance to one that minimizes total within-group squared distance, but not close to an algorithm that minimizes total within-group distance, total cubed within-group distance, or total square-root within-group distance. These tests were again conducted with random datasets and purposefully chosen toy datasets. Note that even when the algorithm is close in performance to these optimizations, it is not always exact for any, and it is not formulated as an optimization problem in general, as Breiger, Boorman and Arabie noted for the CONCOR algorithm [1]. For example, the clustering of the two-dimensional dataset below is not optimal with respect to mean within-group correlations when all reference points are used, where the clusters are denoted by color.

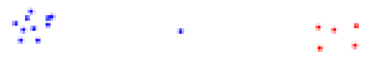

Figure 7: All Reference Points Used

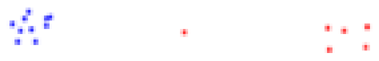

Figure 8: Optimal Clustering for Mean Within-Group Correlations

Note that, though it is the case in the example above, the optimal clustering for mean within-group correlations is not always the same as that of squared within-group distance, and

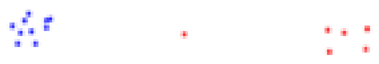

Figure 9: Optimal Clustering for Squared Within-Group Distance

that there may be cases in which one or the other is in fact the same clustering as when all reference points are used with our algorithm.

We also showed that our algorithm is not optimal in minimalizing Shi and Malik's Normalized Cut, or NCut, criterion, and not optimal in minimizing the Cheeger conductance measure used in KVV Spectral Clustering [4]. We also showed that not every point is necessarily in the same cluster as the nearest cluster centroid at the end of the algorithm.

Additionally, simulations were performed on different numbers of dimensions for the data (2, 3 and 4 dimensional Euclidean data), on different data sizes (data sets of 25, 50 and 100 datapoints), and on different methods of choosing the reference points (randomly, farthest points, or closest points), so as to get insight into the amount of reference points needed to get the same result as if all reference points were used (as it was determined that the case of all references points often produced the best clustering in terms of mean within-group correlations). From this, it was learned that for more dimensions and more data points, more reference points were needed to achieve the same clustering as that of all reference points, and that when sampling reference points by using the points that were the farthest apart instead of random points, less reference points were needed to achieve the same clustering. For r points, "farthest points" were selected by ordering the pairwise distances of objects in the system in order from greatest to least, and choosing the unique endpoints of these edges in order. Though more reference points were needed for larger datasets, the proportion of data needed did not exceed half the data points for any randomly generated dataset when averaged over 500 runs for 25, 50, and 100 datapoints (averaged to account for the random generation of reference points used when that was the sampling method being tested).

One explanation as to the benefit of picking the farthest points is that the overall clustering is largely based on the clustering of the reference points, so a reference set that represents the overall distribution well is beneficial. This is due to the fact that the clustering of the reference points does not depend on their relation to any other points, and thus with only the reference points we can already see how these particular points are clustered. After that, we are effectively figuring out in which cluster to place each of the remaining points. This can be easily seen when using the algorithm, as isolating the reference rows of M'_k in fact produces M''_k, all of whose entries have both column and row index in the reference set. Because of this observation, there may be a way to cluster the reference points first, and to intelligently add the remaining points to the resulting clusters. At present, however, we have yet to find an efficient way of doing this, as all relationships of each point to all reference points are used in each step of the algorithm. The reader may hypothesize that we could cluster the reference points and put the values of that M_∞ matrix in the corresponding entries of the M_0 overall matrix before clustering the rest, but that would in fact produce a nonintelligent clustering, as all reference rows in M'_k would be either the same or negative of each other. Similarly, if we have any values in the reference rows and update only the values in the nonreference rows, this does not lead to common convergence in the general case. This is significant as it means any virtual reference points added (points that are used as reference points but that are not in the original dataset) cannot be used as references without themselves being clustered with the dataset, which would in turn lead to a change in the overall structure of the dataset and its clustering.

7 Further Research

The fact that convergence is limited to only the cases we have stated is as of yet still a conjecture and has not been proved in our paper or any of the previous works listed below. McQuitty and

Clark in particular noted the necessity of such a proof and the significance that it could have to this study.

Additionally, we have yet to either prove or disprove whether the two clusters created by this algorithm are always linearly separable. Similarly, a few hypotheses are listed throughout this paper that have yet to be proved (the triangle inequality causing it not be necessary to check M_k for any $k > 0$ and the lack of additional subcases to the different convergence cases).

We also have yet to find an intelligent way to pick the reference points, such that only a constant number is used yet the overall integrity of the clustering is maintained. If such a method was found, it would allow this algorithm to perform at an efficiency of O(ni), as the number of reference points would be constant, which is a substantial improvement for applying this algorithm to larger datasets. More investigation into why our algorithm doesn't always optimize mean within-group correlation when all reference points are used may also aid in this discovery.

In addition, there is still work to be done in investigating what the convergence behavior is like when the reference set is changed infinitely. We know already that undefined values can arise if any reference set causes the next matrix to have undefined values, and that the general case is one of common convergence. However, aside from the exception to common convergence given earlier in this paper, there may still be many other cases, including possibly one in which the matrix oscillates between two sets of values.

Finally, more work has yet to be done in investigating what sort of correlation coefficients are required to have the results detailed above. We know already that the cosine correlation coefficient satisfies common convergence in the general case, but we still need to determine if all of the other results are applicable to the cosine correlation coefficient, and what type of correlation coefficients are needed for our results to be applicable.

8 References

1. Breiger, R., Boorman, S., Arabie, P. (n.d.). An algorithm for clustering relational data with applications to social network analysis and comparison with multidimensional scaling. Journal of Mathematical Psychology, 328-383.

2. Mcquitty, L., Clark, J. (1968). Clusters from Iterative, Interrowar Correlational Analysis. Educational and Psychological Measurement, 28, 211-238

3. Clark, J., Mcquitty, L. (1970). Some Problems and Elaborations of Iterative, Interrowar Correlational Analysis. Educational and Psychological Measurement, 30, 773-784.

4. D. Verma, M. Meila, A comparison of spectral clustering algorithms, Technical Report, Department of CSE University of Washington Seattle, WA, 98195–2350, 2005