Finding Banded Patterns in Data: The Banded Pattern Mining Algorithm

Fatimah B. Abdullahi, Frans Coenen, and Russell Martin

The Department of Computer Science, The University of Liverpool, Ashton Street, Liverpool, L69 3BX, United Kingdom
email: {f.b.abdullahi,coenen,russell.martin}@liverpool.ac.uk

Abstract. The concept of Banded pattern mining is concerned with the identification of "bandings" within zero-one data. A zero-one data set is said to be fully banded if all the "ones" can be arranged along the leading diagonal. The discovery of a banded pattern is of interest in its own right, at least in a data analysis context, because it tells us something about the data. Banding has also been shown to enhances the efficiency of matrix manipulation algorithms. In this paper the exact N dimensional Banded Pattern Mining (BPM) algorithm is presented together with a full evaluation of its operation. To illustrate the utility of the banded pattern concept a case study using the Great Britain (GB) Cattle movement database is also presented.

Keywords: Banded Patterns, Zero-One data, Banded Pattern Mining

1 Introduction

Banded pattern mining is concerned with the identification of "bandings" within zero-one data [7, 10, 8, 9]. The idea is, given a zero-one data set, to rearrange the elements in each dimensions so that a banding is revealed. A zero-one data set is said to be fully banded if all the "ones" can be arranged along the leading diagonal. Typically a perfect banding cannot be discovered, however it is still possible to rearrange the data to reveal a nearest possible banding. The discovery of a banded pattern is of interest in its own right, at least in a data mining context, because it tells us something about the data. Banding has also been shown to improve the operation of various n dimensional matrix manipulation algorithms in that only the portion of the matrix located near the leading diagonal needs to be considered.

Previous work on banding identification [7, 10] has been mostly directed at two dimensional data and has focussed on using heuristics to identify permutations in the data. Two examples are the Minimum Banded Augmentation (MBA) algorithm [8] and the Barycentric (BC) algorithm [9]. These approaches worked well but are difficult to scale up to encompass n dimensional data because of the exponential increase in the number of permutations that need to be considered. The idea proposed in this paper, instead of considering large numbers of permutations, is to use a "banding score" mechanism to iteratively rearrange the elements in each dimension.

The rest of this paper is organised as follows. Section 2 presents some relevant previous work, concentrating on the MBA and BC algorithms. Section 3 presents some formalism concerning the banded pattern problem, while Section 4 presents the proposed BPM algorithm. To aid in the understanding of the BPM algorithm Section 5 presents a worked example. A full evaluation, using both artificial and real data sets, is presented in Section 6. Some conclusion are given in Section 7.

2 Previous Work

The concept of banded matrices, from the data analysis perspective, occurs in many applications domains; examples can be found in paleontology [2], Network data analysis [3] and linguistics [8]. More recent work can be found in [7] and [10]. The current state of the art algorithm, the Minimum Banded Augmentation (MBA) proposed by [8], focuses on minimizing the distance of non-zero entries from the main diagonal of a matrix by reordering the original matrix. The MBA algorithm operates by "flipping" zero entries (0s) to one entries (1s) and vice versa to identify a banding. Gemma et al. fixed the column permutations of the data matrix before executing their algorithm [7]. Here the basic idea is to solve optimally the consecutive one property on the permuted matrix M and then resolve "Sperner conflicts" to eliminate all the overlapping row intervals between each row of the permuted matrix, by going through all the extra rows and making them consecutive. While it can be argued that the fixed column permutation assumption is not a very realistic assumption with respect to many real world situations, heuristical methods were proposed in [7] to determine a suitable fixed column permutation. The MBA algorithm use the Accuracy (Acc) measure to evaluate the performance of the banding produced. Another frequently referenced banding strategy is the Barycentric (BC) algorithm that transposes a matrix. It was originally designed for graph drawing, but more recently used to reorder binary matrices [9]. The BC algorithm uses the Mean Row Moment (MRM) measure to evaluate the performance of the banding produced. The distinction between these previous algorithms and that presented in this paper is that the previous algorithms were all directed at 2D data, while the proposed algorithm operates in ND. The above MBA and BC algorithms are the two examplar banding algorithms with which the operation of the proposed BPM algorithm was compared and evaluated as discussed later in this paper. Note that Bandwidth minimization of binary matrices is known to be NP-Complete [5].

3 Formalism

The data space of interest comprises a set of n dimensions, with $DIM = \{dim_1, dim_2, \ldots, dim_n\}$. Each dimension dim_i comprises a set of sequential coordinates (indexes) commencing with the coordinate 0, $dim_i = \{0, 1, 2 \ldots\}$. Each dimension dim_i also has a set of index positions associated with it $pos_i = \{p_1, p_2, p_3, \ldots\}$. The set of positions is indicated by $POS = \{pos_1, pos_2, \ldots, pos_n\}$.

Note that dimensions are not necessarily all of the same size. However there is a correspondence between each dim_i and pos_i pairing ($|dim_i| \equiv |pos_i|$). Initially the content of each set dim_i will be the same as the content of each set pos_i, however it is the elements in each set pos_i in POS that we wish to rearrange so as to identify a banding.

Each "cell" in the n dimensional data space is thus identifiable by a coordinate tuple of length n, $\langle c_1, c_2 \ldots c_n \rangle$; where $c_1 \in dim_i$, $c_2 \in dim_2$ and so on. If $n = 2$ we might think of these as x-y coordinates, and if $n = 3$ we might think of these as x-y-z coordinates. Each cell contains either a 1 or 0. For illustrative purposes, in the remainder of this paper, we say that a cell with a 1 value contains a "dot", and a cell with a 0 is empty.

The data sets $D = \{d_1, d_2, \ldots\}$ we are interested in thus comprise a sequence of k coordinate tuples each of size n and each reprsenting a "dot". Note that a specific coordinate tuple can appear only once in D. Each tuple thus describes the location of a dot in the data space. It is these dots that we wish to arrange in such a manner that they are as close to the leading diagonal as possible by rearranging the position of the indexes (coordinates).

4 The Banded Pattern Mining Algorithm

A high level view of the proposed Banded Pattern Mining (BPM) algorithm is presented in Algorithm 1. The algorithm describes an iterative approach whereby the elements in each dimension are repeatedly rearranged until a best banding score is arrived at or until some maximum number of iterations is reached. The algorithm is founded on ideas presented in [1] where an approximate BPM algorithm (BPM$_A$) was presented. Comparison are made later in Section 6 using both the proposed exact BPM algorithm and the previous BPM$_A$ algorithm. The inputs to the algorithm are: (i) a zero-one data set D which is to be banded, (ii) a desired maximum number of iterations max and (iii) the set of dimensions DIM describing the n dimensional data space to which D subscribes. The Output is a rearranged data set D'.

On each iteration each dimension dim_i in DIM is considered in turn. For each element j in dim_i a normalised banding score bs_{ij} is calculated (line 12) using equation 2 (the derivation of the equation is described in more detail below). The elements in dimension dim_i are then rearranged (line 14) in descending order according to the banding scores calculated earlier. If two or more elements have the same banding score (an unlikely event given a large data set) then the number of dots featured in each element is taken into consideration together with the relative position of the new location with respect to the centroid of the data space (essentially we wish to place elements with larger numbers of dots closer to the centroid of the current data sub-space than elements with smaller numbers of dots). The rearranging of elements in dimensions is repeated for each dimension in turn.

At the end of each complete iteration the global banding score gbs for the new configuration is determined (line 16) by summing the banding score values derived earlier (Equation 2). If the newly calculated banding score is equal or less (worse) than the previously calculated score we break (line18), otherwise we continue onto the next iteration. On completion (line 25) we derive the new dataset D' using the positions of the indexes contained in DIM. The situation where a worse *gbs* than that obtained on the previous iteration may be arrived at is where we have a “poling” effect where we are rearranging that data one way, and then another way, without improving the banding.

Algorithm 1 The BPM Algorithm

```
 1: Input
 2: $D$ = Binary valued input data set
 3: $max$ = The maximum number of iterations
 4: Output
 5: $D'$ = Data set $D$ rearranged so as to display as near a banding as possible
 6: $gbs = 0$ (The global banding score so far)
 7: $counter = 0$
 8: while $counter < max$ do
 9:    for all $dim_i \in DIM$ do
10:       for $j = 1$ to $j = |dim_i|$ do
11:          $bs_{ij}$ = Banding score for element $j$ in $|dim_i|$ calcuated using
12:               Equation 2
13:       end for
14:       $dim'_i$ = The set $dim_i$ rearranged according to the calculated $bs$
15:            (in descending order)
16:    end for
17:    $gbs'$ = Global banding score for current configuration described by
18:          $DIM' = \{dim'_1, dim'_2, \ldots, dim'_n\}$ calculated using Equation 3
19:    if $gbs' \leq gbs$ then
20:       break
21:    else
22:       $gbs = gbs'$
23:       $DIM = DIM'$
24:    end if
25:    $counter = counter + 1$
26: end while
27: $D'$ = Input dataset $D$ rearranged according to index positions in $POS$
```

The banding score bs_{ij} for a particular element j in dimension dim_i is determined according to the location of the subset of dots S in D whose c_i coordinate is equal to j (recall that each dot in D is define by a coordinate tuple of the form $\langle c_1, c_2 \ldots c_n \rangle$). For each dot in S we calculate the distance to the origin of the data sub-space that does not include dim_i. We exclude the current dimension because this is the dimension we want to rearrange. Thus a banding score bs is calculated as follows:

$$bs = \sum_{i=1}^{i=|S|} distToOrigin(dot_i) \tag{1}$$

However, to allow for comparison of bs we need to normalise the score. Given a dot that is at the origin of the sub-space of interest the normalised bs should be 0. Given a dot that is as far away from the the origin of the sub-space of interest as is geometrically possible the normalised bs should be 1. Thus to normalise bs we need to divide by the sum of the set of $|S|$ maximum distances that can be attained in the given sub space:

$$bs = \frac{\sum_{i=1}^{i=|S|} distToOrigin(dot_i)}{\sum_{i=1}^{i=|M|} max_i} \tag{2}$$

where M is a set of maximum distances corresponding to the number of dots in S. Note that the content of M will vary according to the nature of the set DIM for a given data set D. Given that we can identify the coordinate value that features most frequently in the coordinate tuples in D we know the maximum required size of max. Given our knowledge of DIM we can therefore calculate the values to be included in M at the start of the process (not shown in Algorithm 1) and thus we can calculate these values in advance and store them in a table to be used as necessary. In the evaluation section presented later in this paper a comparison is presented between using a pre calculated BMP "M-table" and calculating maximum distances as required (not using a BPM"M-table"). Using an BPM M-table means that values are only calculated once, although it may be the case that some values are calculated that are never used.

The global banding score, gbs for a configuration is then given by:

$$gbs = \frac{\sum_{i=1}^{i=|DIM|} \sum_{j=1}^{j=|dim_i|} bs_{ij}}{\sum_{k=1}^{k=|DIM|} |dim_k|} \tag{3}$$

the sum of all the identified banding scores for each element in each dimension. Thus if every element within a given data space is filled with a dot the global banding score will be 1. Distances can be calculated in a variety of ways. Two obvious mechanisms are Manhattan distance and Euclidean distance (both are considered in the evaluation presented later in this paper).

5 Worked Example

The operation of the BPM algorithm can best be illustrated using a worked example. Consider the 2 dimensional 4×4 configuration given in Figure 1(a) (the origin is in the top left hand corner). It has dimensions $Dim = \{x, y\}$, and:

$$D = \{\langle 0,1\rangle, \langle 0,2\rangle, \langle 0,3\rangle, \langle 1,0\rangle, \langle 1,1\rangle, \langle 2,0\rangle, \langle 2,1\rangle, \langle 2,3\rangle, \langle 3,2\rangle, \langle 3,3\rangle\}.$$

Considering dimension x first, we calculate banding scores as shown in Table 1. This produces the banding scores 1.00, 0.20, 0.67 and 1.00. We thus rearrange the elements in x in ascending order of their banding scores. Note that in the case where two elements have the same score we arrange things so that the element

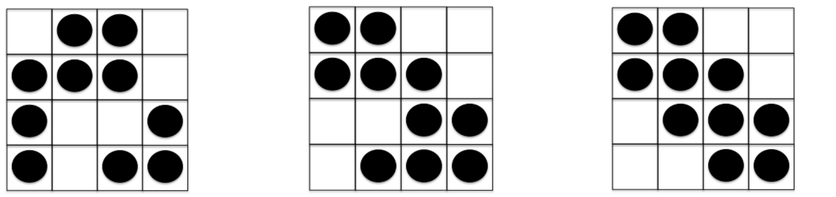
(a) Raw Data (b) After rearrangement of dim_1 (c) After rearrangement of dim_2

Fig. 1. Example of operation of BPM algorithm

# Element	Dist from origin	Max. dist. from origin	bs
0	$1+2+3=6$	$1+2+3=6$	1.00
1	$0+1=1$	$2+3=5$	0.20
2	$0+1+3=4$	$1+2+3=6$	0.67
3	$2+3=5$	$2+3=5$	1.00
		Total	2.87

Table 1. Calculation of banding scores for dimension x

with the largest number of dots associated with it is nearest to the centre of the data space. The result is as shown in Figure 1(b).

Considering dimension y next, we calculate banding scores as shown in Table 2. This produces the banding scores 0.20, 0.67, 1.00 and 1.00. The elements in y are more or less already in ascending order of bs. We only need to swap the last two elements so that the element with the greater number of dots is nearer the centre of the data space. The result is as shown in Figure 1(c).

# Element	Dist from origin	Max. dist. from origin	bs
0	$0+1=1$	$2+3=5$	0.20
1	$0+1+3=4$	$1+2+3=6$	0.67
2	$2+3=5$	$2+3=5$	1.00
3	$1+2+3=6$	$1+2+3=6$	1.00
		Total	2.87

Table 2. Calculation of banding scores for dimension y

The global banding score is then the sum of the individual banding scores divided by the total number of elements in the configuration:

$$gbs = \frac{2.87+2.87}{8} = \frac{5.73}{8} = 0.72$$

The process is repeated on the next iteration (not shown here) and the same gbs value produced because we already have a best banding. The rearranged data set D' arrived at is:

$$D' = \{\langle 0,0\rangle, \langle 0,1\rangle, \langle 1,0\rangle, \langle 1,1\rangle, \langle 1,2\rangle, \langle 2,1\rangle, \langle 2,2\rangle, \langle 2,3\rangle, \langle 3,2\rangle, \langle 3,3\rangle\}.$$

6 Evaluation

This section presents an evaluation and discussion of the proposed exact BPM algorithm. The objectives of the evaluation were as follows:

– To determine the effect of data set size and density on the BPM algorithm using artificial data.
– To compare the operation of the BPM algorithm with existing algorithms (MBA and BC) using real data sets.
– To consider the application of the BPM algorithms with respect to a large scale application (the GB cattle movement database).

Each is discussed in further detail in the following three sub-sections.

6.1 Effect of Data Set Size

To determine the efficiency of the proposed BPM algorithm in comparison with the MBA and BC all three algorithms were run using artificial data sets of varying size generated using the LUCS-KDD data generator [6][1]. Using this generator ten data sets measuring 100×100, 141×141, 173×173, 200×200, 224×224, 245×245, 265×265, 283×283, 300×300 and 316×316 were generated, corresponding to numbers of elements approximately increasing from $10,000$ to $100,000$ in steps of $10,000$. A density of 10% was used (in other words 10% of the cells in each row contained a dot). A further five 100×100 data sets were generated using densities from 10% to 50% increasing in steps of 10.

The recorded runtime results obtained by applying the proposed BPM algorithm and the MBA and BC algorithms to data sets of increasing size are presented in graph form in Figure 2. From the graph it can be seen that there is a clear correlation between the dataset and the run-time, as the dataset size increases the processing time also increases (this is to be expected).

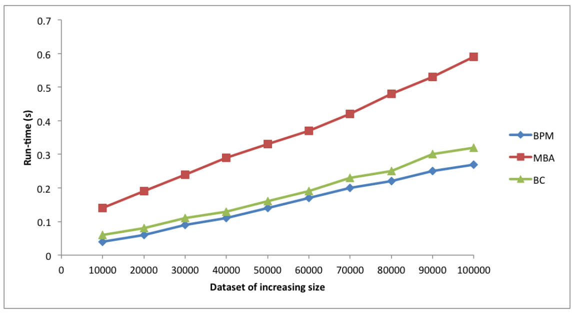

Fig. 2. Recorded run time (sec.) when the BPM, MBA and BC algorithms are applied to data sets of increasing size ($10,000$ to $100,000$ elements in steps of $10,000$)

The recorded runtime results obtained by applying the proposed BPM algorithm and the MBA and BC algorithms to data sets of increasing density are presented in Figure 3. From the graph it can be seen that there is a correlation between the density of the datasets and the run-time as the density of the datasets increases the processing time also increases.

[1] Available at `http://cgi.csc.liv.ac.uk/frans/KDD/Software/LUCS-KDD-DataGen/generator.html`

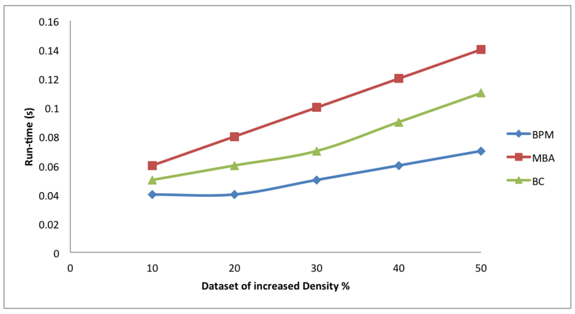

Fig. 3. Recorded run time (sec.) when the BPM, MBA and BC algorithms are applied to data sets of increasing density (10% to 50% elements in steps of 10)

6.2 Comparison with BPM, MBA and BC

To compare the nature of the bandings produced using BPM, MBA and BC the average width of the banding produced was used as an independent measure (as oppose to global banding score gbs, accuracy Acc and Mean Row Moment MRM). Average Banding Width (ABW) was calculated as shown in Equation 4. Similarly, Acc and Mean Row Moment (MRM) are calculated as shown in Equations 5 and 6:

$$ABW = \frac{\sum_{i=1}^{i=|D|} distance\ d_i\ from\ leading\ diagonal}{|D|} \tag{4}$$

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{5}$$

$$MRM = \frac{\sum_{j=1}^{n} j a_{ij}}{\sum_{j=1}^{n} a_{ij}} \tag{6}$$

where TP is the number of "true positives" corresponding to original 1 entries, TN is the number of "true negatives" corresponding to original 0 entries, FP is the number of "false positives" corresponding to transformed 0 entries, FN is the number of "false negatives" corresponding to transformed 1 entries, a_{ij} is the jth entry in row (resp. column) i and n is the number of columns (resp. rows). For the experiments to compare the nature of the bandings produced we applied the algorithms to a range of data sets taken from the UCI machine learning data repository [4]. As in the case of the artificial data sets the UCI data sets are all two dimensional with one dimension representing records and the other data attributes. The data sets were discretised using the LUCS-KDD algorithm [4]. For each dataset we applied the three algorithms, for each algorithm we recorded four scores (i) ABW, (ii) gbs, (iii) Acc and (iv) MRM. Tables 3 and 4 show the results obtained (best scores highlighted using bold font). In terms of the independent ABW metric (and the author's gbs metric) the tables clearly demonstrate that the proposed BPM algorithm outperformed the MBA and BC algorithms. Similarly, in terms of Acc and MRM Table 4 shows that in 6 out of the 10 cases, the proposed BPM algorithm still outperformed MBA and BC. Figures 4, 5 and 6 show the bandings obtained using the glass data set and the

BPM, MBA and BC algorithms respectively. The Figures shows that banding can be identified in all cases. However, considering the banding produced when the MBA algorithm was applied to the glass data sets (Figure 5) the banding result contains dots ("1s") at top-right and bottom-left corners while the BPM algorithm does not. Similarly, when the BC algorithm was applied to the glass data set (Figure 6) the banding result is less dense than in the case of the proposed BPM algorithm (which features a smaller bandwidth).

Table 3. *ABW* and *gbs* results presented using: *BPM*, *MBA* and *BC* algorithms

Dataset	(column	BPM	MBA	BC	BPM	MBA	BC
	× rows)	*ABW*	*ABW*	*ABW*	*gbs*	*gbs*	*gbs*
Auto	205	**0.6125**	0.8182	0.7802	**0.7604**	0.6545	0.6109
Breast	699	**0.9852**	0.9982	0.9957	**0.8890**	0.7281	0.7423
Car	1728	**0.9916**	0.9983	0.9973	**0.8053**	0.7783	0.7697
Congress	435	**0.9588**	0.9918	0.9881	**0.8807**	0.8086	0.8018
Cylband	540	**0.8913**	0.9659	0.9496	**0.8405**	0.7854	0.7417
Dematology	366	**0.9205**	0.9729	0.9709	**0.8189**	0.7742	0.7553
Ecoli	336	**0.9149**	0.9908	0.9717	**0.7767**	0.7544	0.7697
Flare	1389	**0.9794**	0.9981	0.9924	**0.8014**	0.7379	0.7807
Glass	214	**0.8392**	0.9468	0.9391	**0.7744**	0.7503	0.6963
Ionosphere	351	**0.7152**	0.8295	0.8696	**0.7906**	0.7393	0.6882

Table 4. *Acc* (%) and *MRM* results presented using: *BPM*, *MBA* and *BC* algorithms

Dataset	(column	BPM	MBA	BC	BPM	MBA	BC
	× rows)	*Acc*	*Acc*	*Acc*	*MRM*	*MRM*	*MRM*
Auto	205	71.47	**73.37**	73.35	**135.64**	129.78	130.81
Breast	699	**52.62**	51.25	51.25	**437.72**	358.75	379.13
Car	1728	61.97	**62.01**	61.13	1168.23	1146.58	**1171.21**
Congress	435	55.52	**55.68**	**56.34**	**352.21**	348.66	322.72
Cylband	540	62.86	**63.42**	63.32	**352.21**	348.66	322.72
Dematology	366	**65.70**	65.05	61.79	242.99	238.17	**245.77**
Ecoli	336	**61.94**	60.36	60.32	**254.70**	239.50	249.17
Flare	1389	**63.91**	63.27	63.21	**1051.57**	1031.53	1015.01
Glass	214	**74.88**	72.68	72.86	149.84	141.35	**150.84**
Ionosphere	351	**66.17**	65.94	65.90	**244.55**	243.76	230.42

6.3 Large scale study

To illustrate the utility of the proposed BPM algorithm and the previous BPM$_A$ algorithm, the authors have applied both algorithms to a 5 dimensional data set constructed from the GB Cattle movement data base. The GB cattle movement database records all the movements of cattle registered within or imported into Great Britain. The database is maintained by the UK Department for Environment, Food and Rural Affairs (DEFRA). For the analysis reported in this work, data for the year 2003, for four counties (Aberdeenshire, Cornwall, Lancashire and Norfolk in Great Britain). In total we generated 16 data sets. The

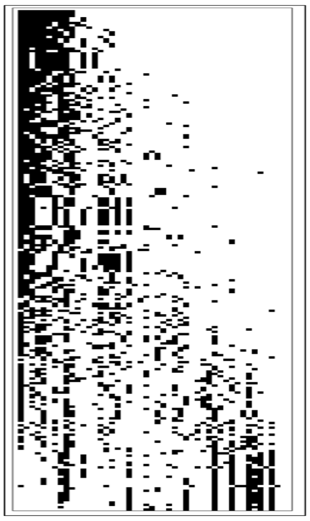
Fig. 4. BPM banding resulting using Glass dataset

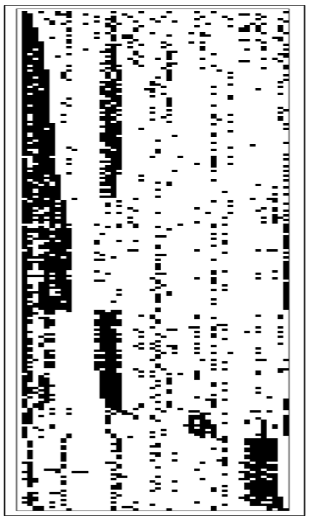
Fig. 5. MBA banding resulting using Glass dataset

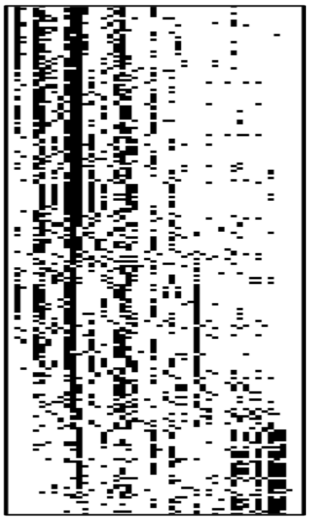
Fig. 6. BC banding resulting using Glass dataset

easting and northing dimensions were divided into ten sub-ranges and the temporal dimension divided into 3 intervals (each interval represented a month) was used. Each record comprises: (i) Animal Gender, (ii) Animal age, (iii) the cattle breed type, (iv) sender location in terms of easting and northing grid values, (v) the type of the sender location, (vi) receiver location type and (vii) the number of cattle moved. Discretization and Normalization were undertaken using the LUCS-KDD ARM DN Software[2] to convert the input data into the desired zero-one format. As a result the GB dataset comprised 110 items distributed over five dimensions: records, attributes, eastings, northings and time (months). Some statistics concerning the data sets are presented in Table 5.

The results obtained are presented in Table 6 (best results highligted in bold font). Note that the table includes results from using and not using a BPM M-table (as considered in Section 4) and results using the BPM$_A$ algorithm presented previously in [1]. From the table the following can be noted: (i) using the BPM$_A$ algorithm is more efficient than using BPM (Euclidean and Manhattan) (ii) not using BPM M-table requires less runtime than when using such a table (iii) using BPM with Manhattan weighting is more efficient than when using BPM with Euclidean weighting (because the use of Manhattan distances entail less calculation) and (iv) using BPM with Euclidean weighting produces the best bandings (*gbs* scores). Thus we can conclude that BPM with Euclidean weighting coupled with the use of a M-table is the most effective approach to the banded pattern mining problem. Closer inspection of the table also indicates that, as expected, there is a correlation between the number of records in the

[2] http://www.csc.liv.ac.uk/∼frans/KDD/Software/LUCS_KDD_DN_ARM.

data sets and the run-time; as the number of records increases the processing time increases (this is to be expected). Note that the BPM$_A$ algorithm does not necessarily find a best banding but only an approximation because it does not consider the entire data space when calculating bandings (it only consider dimension pairings). The BPM algorithm presented in this paper is designed to find an exact banding, instead of considering only dimension pairings as presented in [1], the bandings are derived with respect to the entire data space.

Table 5. Number of items in each dimension (after discretization) for the 16 5-D GB cattle movement data sets

Counties	Years	# Recs.	# Atts.	# Eings.	# Nings.	# Time
Aberdeenshire	Abd-Q1	42962	98	10	10	3
	Abd-Q2	46187	101	10	10	3
	Abd-Q3	41181	104	10	10	3
	Abd-Q4	47842	107	10	10	3
Cornwall	Corn-Q1	40501	101	10	10	3
	Corn-Q2	39626	104	10	10	3
	Corn-Q3	40226	107	10	10	3
	Corn-Q4	49890	110	10	10	3
Lancashire	Lanc-Q1	34325	97	10	10	3
	Lanc-Q2	40926	100	10	10	3
	Lanc-Q3	45765	103	10	10	3
	Lanc-Q4	47392	106	10	10	3
Norfolk	Nolf-Q1	11526	98	10	10	3
	Nolf-Q2	14311	101	10	10	3
	Nolf-Q3	9460	104	10	10	3
	Nolf-Q4	11680	107	10	10	3

7 Conclusion

In this paper we have presented the BPM algorithm. Unlike the existing MBA and BC algorithms, the proposed algorithm does not consider large numbers of permutations but instead uses the concept of banding scores. In addition the proposed mechanism operates in N-D. The results presented indicate that the proposed BPM algorithm produces a better banding, using an independent Average Banding Width (ABW) measure, than in the case of MBA and BC. Results were also presented, using a large scale study (directed at the GB cattle movement database), indicating that the BPM algorithm and the BPM$_A$ algorithm (a previous variation of the proposed BPM algorithm) work well in 5-D (in terms of efficiency and effectiveness). The BPM mining algorithm is also more efficient than MBA and BC because it does not have to consider large numbers of permutations. For future work the authors intends to investigate High Performance Computing variation of the BPM algorithm to allow it to be applied in the context of Big Data Analytics.

Table 6. Runtime (*RT*) and *gbs* results obtained using: (i) Manhattan and Euclidean BPM and no M-Table (ii) Manhattan and Euclidean BMP and M-Table and (iii) BPM$_A$

		runtime (sec)					Global banding		
Month	#	BPM no M Tab.		BPM and M Tab.		BPM$_A$	*gbs* score		
ID	Recs.	Manhat.	Euclid.	Manhat.	Euclid.		Manhat.	Euclid.	BPM$_A$
Abd-Q1	42962	**15.66**	48.68	**61.13**	212.03	**10.95**	0.6854	**0.6937**	0.4115
Abd-Q2	46187	**19.82**	50.95	**60.01**	219.04	**16.95**	0.6861	**0.6939**	0.4142
Abd-Q3	41181	**30.29**	43.86	**58.32**	211.89	**08.83**	0.6892	**0.6942**	0.4152
Abd-Q4	47842	**28.01**	80.86	**32.22**	231.59	**16.44**	0.6867	**0.6932**	0.4101
Corn-Q1	40501	**28.73**	93.82	**45.35**	163.42	**07.83**	0.6880	**0.6934**	0.4190
Corn-Q2	39626	**20.32**	69.33	**51.60**	185.91	**06.92**	0.6834	**0.6931**	0.4239
Corn-Q3	40226	**33.13**	94.41	**67.87**	201.86	**07.58**	0.6840	**0.6944**	0.4263
Corn-Q4	49890	**48.92**	121.11	**80.59**	210.54	**18.88**	0.6885	**0.6943**	0.4221
Lanc-Q1	34325	**27.66**	46.68	**51.02**	147.29	**05.13**	0.6856	**0.6936**	0.4346
Lanc-Q2	40926	**36.50**	50.95	**63.11**	182.74	**09.91**	0.6860	**0.6938**	0.4350
Lanc-Q3	45765	**25.74**	52.03	**59.85**	204.87	**13.86**	0.6859	**0.6936**	0.4352
Lanc-Q4	47392	**36.29**	55.52	**80.52**	228.89	**15.99**	0.6854	**0.6936**	0.4368
Nolf-Q1	11280	**05.32**	26.70	**19.65**	46.21	**01.58**	0.6830	**0.6934**	0.4124
Nolf-Q2	14557	**17.04**	25.85	**47.40**	86.82	**02.29**	0.6814	**0.6937**	0.4139
Nolf-Q3	9460	**10.48**	22.23	**45.17**	56.20	**01.27**	0.6852	**0.6942**	0.4202
Nolf-Q4	11680	**13.34**	25.84	**46.38**	63.15	**02.23**	0.6820	**0.6939**	0.4133

References

1. F. B Abdullahi, F Coenen, and R. Martin. A scalable algorithm for banded pattern mining in multi-dimensional zero-one data. In *In Proc. Data Warehousing and Knowledge Discovery (DaWaK'14). Springer, LNAI*, pages 391–404, 2014.
2. J. Atkins, E. Boman, and B. Hendrickson. Spectral algorithm for seriation and the consecutive ones problem. *SIAM J. Comput.*, 28:297–310, 1999.
3. A. Banerjee, C. Krumpelman, J. Ghosh, S. Basu, and R. Mooney. Model-based overlapping clustering. In *Proceedings of Knowledge Discovery and DataMining*, pages 532–537, 2005.
4. F. Coenen. LUCS-KDD ARM DN software. http:// www.csc.liv.ac.uk / frans/KDD /Software / LUCS_KDD_DN_ARM/, 2003.
5. A. E. Cuthill and J. McKee. Reducing bandwidth of sparse symmetric matrices. In *Proceedings of the 1969 29th ACM national Conference*, pages 157–172, 1969.
6. C. Frans. LUCS-KDD Data generator software. http:// www.csc.liv.ac.uk / frans /KDD /Software / LUCS_KDD_DataGen_Generator.html, 2003.
7. G .C Gemma, E. Junttila, and H. Mannila. Banded structures in binary matrices. *Knowledge Discovery and Information System*, 28:197–226, 2011.
8. E. Junttila. *Pattern in Permuted Binary Matrices.* PhD thesis, 2011.
9. E . Makinen and H. Siirtola. The barycenter heuristic and the reorderable matrix. *Informatica*, 29:357–363, 2005.
10. H. Mannila and E. Terzi. Nestedness and segmented nestedness. In *Proceedings of the 13h ACM SIGKDD international conference on knowledge discovery and data mining, New York, NY, USA, 2007*, pages 480–489, 2007.