Multi-Dimensional Banded Pattern Mining

Fatimah B. Abdullahi[1] and Frans Coenen[2]

1. Department of Computer Science, Ahmad Bello University, Zaria Kaduna state, Nigeria;
2. Department of Computer Science, The University of Liverpool, Liverpool, UK.
fbabdullahi1@yahoo.com, coenen@liverpool.ac.uk

Abstract. Techniques for identifying "banded patterns" in n-Dimensional (n-D) zero-one data, so called Banded Pattern Mining (BPM), are considered. Previous work directed at BPM has been in the context of 2-D data sets; the algorithms typically operated by considering permutations which meant that extension to n-D could not be easily realised. In the work presented in this paper banding is directed at the n-D context. Instead of considering large numbers of permutations the novel approach advocated in this paper is to determine *banding scores* associated with individual indexes in individual dimensions which can then be used to rearrange the indexes to achieve a "best" banding. Two variations of this approach are considered, an approximate approach (which provides for efficiency gains) and an exact approach.

Keywords: Banded Patterns in Big data, Banded Pattern Mining.

1 Introduction

The work presented in this paper is concerned with techniques for identifying "banded patterns" in n-Dimensional (n-D) binary valued data sets, data sets which comprise only ones and zeroes. For ease of understanding, in this paper, the presence of a one is conceptualised as a "dot" (a sphere in 3-D and a hypersphere in n-D, although the term dot will be used regardless of the number of dimensions considered). The presence of a zero is then indicated by the absence of a "dot". The objective is to rearrange the indexes in the individual dimensions so that the dots are arranged along the leading diagonal of the matrix representing the data, or as close to the leading diagonal as possible.

Binary valued data occurs frequently in many real world application domains, examples include bioinformatics (gene mapping and probe mapping) [6, 16, 27], information retrieval [10] and paleontology (sites and species occurrences) [7, 18]. The advantages offered by banding data are fourfold:

1. The resulting banding may be of interest in its own right in that it tells us something about the data; for example it shows us groupings (clusterings) of records that co-occur.
2. Following on from (1) banding may enhance our interpretability of the data; providing insights that were not clear before.
3. It also allows for the visualisation of the data, which may enhance our understanding of the data.

4. It serves to compress the data, which may consequently enhance the operation of algorithms that work with zero-one data.

The banding of zero-one data in 2-D has a long history; however the idea of Banded Pattern Mining (BPM), as conceived of in this paper, was first proposed in [24, 26] (see also [3, 20, 21]). This early work on BPM was focussed on the heuristically controlled generation and testing of permutations. The generation of permutations is known to be an NP-complete problem, thus the algorithms presented in [24] and [26] were not easily scalable, hence they were directed at 2-D data. The work presented in this paper is directed at finding bandings in n-Dimensional (n-D) data using the concept of banding scores, an idea first presented in [1, 2]. More specifically using a BPM algorithm that uses the banding score concept to assign banding scores to individual indexes in individual dimensions, and reordering the index dimensions accordingly. Two variations of this algorithm are considered, exact and approximate, founded on preliminary work presented in [4, 5]. An overall measure of the quality of a banding featured in a given dataset can then be obtained by combining and normalising the individual banding scores to give a Global Banding Score (GBS).

The rest of this paper is structured as follows. Section 2 presents a brief background review of some relevant work. Section 3 then gives the BPM formalism. Section 4 presents the proposed banding score calculation mechanism, while Section 5 presents the BPM algorithms that utilise the proposed approximate and exact banding score mechanisms. The evaluation of the proposed approaches is presented in Section 6. The paper is concluded in Section 7 with a review of the main findings.

2 Related Work

The work presented in this paper is directed at effective banding mechanisms that operate with n-D zero-one data sets. Example applications include: network analysis [19], co-occurrence analysis [18], VLSI chip design [22] and graph drawing [28]. In the case of network analysis the objective is typically community detection. To apply banding the network of interest needs to be represented in the form of an adjacency matrix. By rearranging the rows and columns of the adjacency matrix a banding can be obtained that features groupings of nodes which in turn will be indicative of communities. A specific example can be found in [19] where an American football network data set was used; the communities of interest were teams that frequently played each other. In co-occurrence analysis the aims is the identification of entities that *co-occur*. A specific example can be found in [18] where a paleontological application was considered; here the aim was to match up Neolithic sites with fossil types. The application of banding in the context of VLSI chip design is concerned with the “block alignment problem”, where banding allows for the identification of “channels ” between the circuit component blocks [22]. In the case of graph drawing we wish to minimise the number of “edge cross overs”, this can also be identified using the banding concept [28].

The concept of banded data has its origins in numerical analysis [8] where it has been used in the context of the resolution of linear equations. The banding concept is also related to the domain of reorderable matrixes, reorderable patterns and bandwidth minimisation. Reorderable matrices are concerned with mechanisms for visualising (typically) 2-D tabular data so as to achieve a better understanding of the data [11, 12]. The idea of reorderable matrices dates back to the 19th century when Petrie, an English Egyptologist, applied a manual reordering technique to study archaeological data [23, 25]. Since then a number of reordering methods have been proposed with respect to a variety of applications. Of note with respect to the work presented in this paper is the BC algorithm [24] which was originally proposed to support graph drawing. The BC algorithm is directed at 2-D data and operates by finding permutations for both rows and columns, such that non-zero entries are as close to each other as possible using a barycentric measure describing the average position of dots in a given column/row. The significance of the BC algorithm with respect to this paper is that it is used as a comparator algorithm with which to evaluate the banding techniques presented.

Reorderable patterns are akin to reorderable matrices, however the idea is to reorder columns and rows so as to reveals some (hidden) pattern of interests [9, 17], as opposed to providing a means of facilitating data visualisation. As such the motivation for reorderable patterns can be argued to be the same as that for BPM; the distinction is that the idea of reorderable patterns is concerned with any pre-prescribed pattern P that can be revealed by reordering the columns and rows in a 2-D matrix not just banding (it is also not necessarily directed at zero-one data). This can be viewed as a generalisation of the BPM problem in the sense that the patterns we are looking for in BPM are comprised of dots arranged about the leading diagonal. P in this case would be the locations about the leading diagonal up to a certain distance away, however in the context of the domain of reorderable patterns P can be any shape. The challenge of reorderable patterns is finding a permutation by which the pattern P is revealed.

Bandwidth minimisation [14], [15] is concerned with the process of minimizing the bandwidth of the non-zero entries of a sparse 2-D matrix by permuting (reordering) its rows and columns such that the non-zero entries form a narrow "band " that is as close as possible to the leading diagonal. Bandwidth minimisation is clearly also akin to the BPM. The distinction is that bandwidth minimisation is directed at the specific objective of minimising bandwidth to aid further processing of (typically) 2-D matrices, while BPM is directed at data analysis (a by-product of which happens to be bandwidth minimisation and also visualisation).

There has been some limited previous work directed at Banded Pattern Mining (BPM) as conceived of in this paper where BPM is defined as the identification of hidden bandings in zero-one data sets. Of particular note in this context is the work of Gemma et al. [19] who proposed the Minimum Banded Augmentation (MBA) algorithm. The algorithm considers a series of column permutations to produce a number of permuted matrices (ordered matrices). Each column permutation is considered to be fixed whilst row permutations are conducted and evaluated. Two variations of the MBA algorithms have been proposed [19]; the Minimum Banded Augmentation Fixed Permutation (MBA$_{FP}$) algorithm and the Minimum Banded Augmentation Bi-directional Fixed Permutation (MBA$_{BFP}$) algorithm. Both algorithms featured the joint disadvantages of: (i) being computationally expensive; and, as consequence and (ii) of being only applicable to 2-D data. The significance of the MBA$_{FP}$ and MBA$_{BFP}$ algorithms, with respect to the work presented in this paper, is that they were also used to compare the operation of the considered BPM algorithms.

3 BPM Formalism

The data spaces of interest comprise a set of dimensions DIM, where $DIM = \{Dim_1, \ldots, Dim_n\}$. The dimensions are not necessarily of equal size, and each dimension Dim_i comprises a sequence of k index values $\{e_{i_1}, \ldots, e_{i_k}\}$. In 2-D we can conceive of the dimensions as being columns and rows, and in 3-D as columns, rows and slices. Each location may contain zero, one or more dots. The precise distribution of the dots depends on the nature of the application domain. Each dot (hyper-sphere in n-D space) will be represented by a set of coordinates: $\langle c_1, \ldots, c_n \rangle$. The challenge is then to rearrange the indexes in the dimensions so that the dots are arranged along the leading diagonal (or as close to it as possible) taking into consideration that individual locations may hold multiple dots.

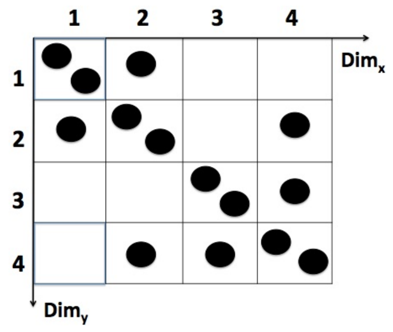

Fig. 1. 2-D multiple dots configuration featuring a banding

4 Calculation of banding Scores

The fundamental idea underpinning the BPM algorithms considered in this paper is the concept of a banding score [1]. The idea is that given a dimension $Dim_i = \{e_{i_1}, \ldots, e_{i_k}\}$ we can calculate a weighting for each index $e_{i_j} \in Dim_i$ according to the dots that feature at that index. To do this we sum the distances of the dots to the origin of a *modified data space* defined by DIM' where DIM' is the set of dimensions in Dim_i excluding the current dimension to be rearranged. Thus if $DIM = \{Dim_1, Dim_2, Dim_3, Dim_4\}$ and we wish to rearrange the indexes in Dim_2 then $DIM' = \{Dim_1, Dim_3, Dim_4\}$. In the case of a 2-D space, where $DIM = \{Dim_x, Dim_y\}$, this would mean calculating the banding sores for Dim_x with respect to Dim_y, and for Dim_y with respect to Dim_x, which would mean simply summing the y (x) index values and normalising by the maximum indexes taking into account the potential for multiple dots. Given a 2-D space $DIM = \{Dim_i, Dim_j\}$ the banding score for index p in dimension i, bs_{i_p}, will be calculated as shown in Equation 1.

$$bs_{i_p} = \frac{\sum_{i=1}^{i=|Index|} index_i \in Index \times m_i \in M}{\sum_{i=1}^{i=|Index|} (k_j - i + 1) \times m_i \in M'} \tag{1}$$

where: (i) $Index = \{index_1, index_2, \ldots, index_k\}$ is a list of Dim_j indexes for dots that feature index p in Dim_i; (ii) $M = \{m_1, m_2, \ldots, m_k\}$ is a list of the number of dots at each location corresponding to list I; (iii) M' is a list similar to M but arranged in descending order according to the number of dots at each location; and (iv) k_j is the size of Dim_j. Referring to the example given in Figure 1 the list I for the first column will be $\{1, 2\}$ (note that the index numbering starts from 1, not 0), the associated list M will be $\{2, 1\}$ and the list M' will also be $\{2, 1\}$. Thus:

$$bs_{x_1} = \frac{(1 \times 2) + (2 \times 1)}{(4 \times 2) + (3 \times 1)} = \frac{4}{11} = 0.363$$

Following on from this $bs_{x_2} = 0.692$, $bs_{x_3} = 0.909$ and $bs_{x_4} = 1.000$. The same scores would be obtained for the y dimension in Figure 1 because the banding is symmetrical about the leading diagonal. The idea is then to reveal a "best" banding by arranging the indexes, in ascending order from the origin, according to their associated banding scores.

Translating the above to address n-D data, banding scores would be calculated as shown in Equation 2 where: (i) C a list of locations in the modified data space (data space of size $n - 1$ where $n = |DIM|$); (ii) Max is a list of maximum distances, arranged in descending order such that $|Max| = |C|$; and (iii) M and M' are defined in the same manner as before. The function $dist()$ returns a distance to the origin of the modified data space from the location of its argument expressed as a set of coordinates of the form $\langle c_1, c_2, \ldots, c_{n-1}\rangle$ ($n - 1$ because the modified data space has one dimension less than the original data space). Distance can be determined in a number of manners but two obvious alternatives are Euclidean distance and Manhattan distance. The derivation of the set Max is not as straightforward as it first seems and is therefore discussed in further detail in Sub-section 5.1. Using Equation 2, banding scores can be calculated for each dimension and used to iteratively rearrange the indexes in the individual dimensions to reveal a banding.

$$bs_{i_p} = \frac{\sum_{i=1}^{i=|C|} dist(c_i \in C) \times m_i \in M}{\sum_{i=1}^{i=|Max|} (max_i \in Max) \times m_i \in M'} \tag{2}$$

The banding score concept can also be used to calculate a Global Banding Score (GBS) for an entire banding configuration using Equation 3 where GBS_i is the the GBS for dimension i (Dim_i).

$$GBS = \frac{\sum_{i=1}^{i=|DIM|} GBS_i}{|DIM|} \tag{3}$$

The value for GBS_i is then calculated using Equation 4. Note that each banding is weighted according to its index location as we wish the diagonal around which our dots are arranged to be from the origin of the data space of interest. This means we have to normalise using $\frac{k_i(k_i+1)}{2}$.

$$GBS_i = \frac{\sum_{p=1}^{p=k_i} bs_{i_p} \times (k_i - p + 1)}{\frac{k_i(k_i+1)}{2}} \tag{4}$$

Thus, returning to the configuration given in Figure 1, using Equation 4, the value for GBS_x will be calculated as follow:

$$\frac{(0.363 \times 4) + (0.692 \times 3) + (0.909 \times 2) + (1.000 \times 1)}{\frac{4(5)}{2}} =$$

$$\frac{1.452 + 2.076 + 1.812 + 1.000}{10} = 0.634$$

The configuration is symmetrical about the leading diagonal, thus GBS_y will also equal 0.634. The GBS for the entire configuration will then, using Equation 3, be:

$$\frac{0.634 + 0.634}{2} = 0.634$$

Note that, with reference to Equation 4, if every cell in a given data space holds exactly the same number of one or more dots, thus no banding at all, the GBS_i for each dimension i will be 1 and the GBS will also be 1. Thus we wish to minimise the GBS value for a configuration to arrive at a best banding. (If the data space contains no dots at all GBS will be 0.).

5 Banded Pattern Mining

The banding score concept, as described above, can be incorporated in BPM algorithms in various ways. Two are presented in this section, Approximate BPM (ABPM) and Exact BPM (EBPM). The first, as the name suggests, produces an approximate (but arguably acceptable) banding while the second, again as the name suggests, produces an exact banding; the advantage of the first is that it is more efficient. At a high level both algorithms work in a similar manner as shown in Algorithm 1. The inputs (lines 1 to 2) are: (i) a dot data set D, comprising a set of tuples of the form $\langle c_1, c_2, \dots \rangle$, describing the location of each dot in the data space; and (ii) the set of dimensions $DIM = \{Dim_1, Dim_2, \dots, Dim_n\}$ associated with D. The output is a rearranged data set D that minimises the GBS value. The algorithm iteratively loops over the data space. On each iteration the algorithm rearranges the indexes in the set of dimensions DIM, using the banding score concept, to produce a revised set of dimensions DIM' (line 6). This revised set of dimensions is then used to rearrange D to give D' (line 7). A new GBS is then calculated (using Equations 3 and 4). Then, if the new GBS (GBS_{new}) is worse than the current GBS (GBS_{sofar}).The algorithm exits with the previously stored configuration and GBS. Otherwise D, DIM and the value for GBS_{sofar} are updated (lines 12 to 14) and the algorithm repeats.

The ABPM variation of the BPM algorithm considers pairings of dimensions, calculating banding scores using Equation 1. The advantage, over the EBPM variation, is that the banding score calculation is much more efficient than when calculated using Equation 2. For the ABPM variation, lines 6 to 8 in Algorithm 1 are replaced with the

Algorithm 1 Generic BPM Algorithm

```
1: Input: $D$ = Zero-one data matrix subscribing to $DIM$,
2:     $DIM = \{Dim_1, Dim_2, \ldots, Dim_n\}$
3: Output: Rearranged data space $D$ that minimise $GBS$
4: $GBS_{sofar} = 1.0$
5: loop
6:     $DIM'$ = The set of dimensions $Dim$ rearranged using either approximate or exact BPM
7:     $D'$ = The data set $D$ rearranged according to $DIM'$
8:     $GBS_{new}$ = The GBS for $D'$ calculated using Equations 3 and 4
9:     if $(GBS_{new} \geq GBS_{sofar})$ then
10:        break
11:    else
12:        $D = D'$
13:        $DIM = DIM'$
14:        $GBS_{sofar} = GBS_{new}$
15:    end if
16: end loop
17: Exit with $D$ and $GBS$
```

Algorithm 2 The ABPM Variation

```
1: for $i = 1$ to $i = |DIM| - 1$ do
2:     for $j = i + 1$ to $j = |DIM|$ and $j \neq i$ do
3:         for $p = 1$ to $p = |K_i|$ do
4:             $bs_{ij_p}$ = Banding score for index $p$ in $Dim_i$ calculated w.r.t. $Dim_j$ using
                   Equation 1
5:         end for
6:         $DIM' = Dim_i$ rearranged according to $bs_{ij}$ values
7:         $D' = D$ rearranged according to $DIM'_i$
8:     end for
9: end for
```

pseudo code given in Algorithm 2. With reference to Algorithm 2, ABPM operates by considering all possible dimension pairings ij. For each pairing the banding score bs_{ij_p} for each index p in dimension Dim_i is calculated with respect to Dim_j (line 4). The calculated banding score values are then used to rearrange the indexes in dimension Dim_i (line 6) and consequently the data space D (line 8).

For the EBPM variation lines 6 to 7 in Algorithm 1 are replaced with the pseudo code given in Algorithm 3. As in the case of the ABPM variation, the EBPM algorithm iteratively loops over the data space calculating banding scores for each index p in each dimension Dim_i. For each dimension, the bs_{i_p} values are used to rearrange the indexes in the dimension (line 6 in Algorithm 1) which is then used to reconfigure D. Inspection of Algorithm 1 indicates that Equation 2 is called repeatedly and on each occasion a set Max will be generated. It therefore makes sense to generate a collection of Max sets in advance and store these in a Maximum set Table (an M-Table). Each row in this table will represent one of the n dimensions to be rearranged. The length of each row will be equivalent to the largest number of dots associated with a single index in the dimension associated with the row.

Algorithm 3 The EBPM Variation

```
1: for $i = 1$ to $i = |DIM|$ do
2:     for $p = 1$ to $p = k_i$ do
3:         $bs_{i_p}$ = Banding score for index $p$ in $Dim_i$ calculated using Equation 2
4:     end for
5:     $DIM' = Dim_i$ rearranged according to $bs_i$ values
6:     $D' = D$ rearranged according to $DIM'_i$
7: end for
```

5.1 Generation of the set Max

In the foregoing, Equation 2 requires a set Max, a set of maximum distances of potential dot locations to the origin. The size of the set max depends on the number of dot locations to be considered (the size of the set C in Equation 2). The calculation of the longest possible distance from the origin to a dot within a n-D space is straight forward as the maximum coordinates are known. The second most longest distance is harder, especially where the ND space under consideration is not symetrical. Similarly with the third longest distance and so on. Other than for the maximum distance there will be a number of candidates locations that will give the nth most longest distance.

An algorithm for populating the set Max is thus given in Algorithm 4 (the Maximum Distance Calculation, or MDC, algorithm). The inputs to the algorithm are: (i) the number of maximum values to be returned (thus the size of the desired set Max) and (ii) the dimension sizes ($n-1$ because we exclude the current dimension for which banding scores are being calculated). The output is a list of maximum distances in descending order. On start up the location loc_1, which will feature the maximum distance, is identified and stored in the set Loc (line 5). The associated distance $dist_1$ is then calculated and stored in the set $Dist$ (line 7). The algorithm then continues, in an iterative manner, according to the $numValues$ input parameter. On each iteration the longest distance in $Dist$ is extracted and added to the list Max (lines 9 and 10). This distance is then pruned from the set $Dist$ (line 11) and the associated location pruned from the set Loc (line 12). The algorithm then (lines 13 to 18) calculates a new set of locations, to be added to Loc, by iteratively subtracting 1 from each coordinate associated with Loc_j in turn ($loc_j = \langle c-1, c_2, \ldots, c_{n-1} \rangle$) and thus creating new location adjacent to loc_j. If not already in Loc each new location loc is appended to the set $NewLoc$. The algorithm then used the set $NewLoc$ to calculate a new set of distances $NewDist$ (lines 20 to 24). The sets $NewLoc$ and $NewDist$ are then appended to the existing sets Loc and $Dist$ and the process repeated until there are no more maximum distances to calculate. Note that the function $dist()$ in Algorithm 4 is the same as that used in Equation 2.

Algorithm 4 Maximum Distance Calculation (MDC) Algorithm

```
1: Input: $numValues$ = The size of the desired set $Max$,
   $DimSizes = \{k_1, k_2, \dots k_{n-1}\}$ (The sizes of the dimensions to be considered)
2: Output: $Max$ = The desired set of maximum distances
3: $Max = \emptyset$
4: $loc_1 = \langle k_1, k_2, \dots k_{n-1} \rangle$
5: $Loc = \{loc_1\}$ (Running list of locations)
6: $dist_1 = dist(loc_1)$
7: $Dist = \{dist_1\}$ (Running list of distances)
8: for $i = 1$ to $i = numValues$ do
9:     $j = getLongestDistIndex(Dist)$
10:    $Max = Max \cup Dist_j$
11:    $Dist = Dist - dist_j$ (Prune $dist_j$ from $Dist$)
12:    $Loc = Loc - loc_j$ (Prune $loc_j$ from $Loc$)
13:    $NewLoc = \emptyset$
14:    for $q = 1$ to $q = |DimSizes|$ do
15:        $loc = loc_j$ with $c_i$ replaced with $c_i - 1$
16:        if $loc \notin Loc)$ then
17:            $NewLoc = NewLoc \cup loc$
18:        end if
19:    end for
20:    $NewDist = \emptyset$
21:    for $q = 1$ to $q = |NewLoc|$ do
22:        $dist = dist(newLoc_q \in NewLoc)$
23:        $NewDist = NewDist \cup dist$
24:    end for
25:    $Loc = Loc \cup NewLoc$
26:    $Dist = Dist \cup NewDist$
27: end for
```

6 Evaluation

This section presents the evaluation of the BPM algorithms considered in this paper. All the reported experiments were conducted using either: (i) 2-D data sets available within the UCI data mining repository [13] or (ii) 5-D data sets extracted from the Great Britain (GB) Cattle Tracking System (CTS). The latter was selected because it can be interpreted in the context of five dimensions. The objectives of the evaluation was firstly to compare the operation, in terms of the overall GBS and runtime (seconds), of the ABPM and EBPM algorithms, using both the Euclidean and Manhattan variations of the latter, in the context of n-D data sets (specifically 5-D data sets); and secondly to compare the operation, using an independent Average Band Width (ABW) measure as well as the overall GBS obtained, and runtime (in seconds), of the best BPM algorithm from (1) with the previously proposed BC and MBA algorithms.

Table 1. Comparison of BPM algorithms in terms of GBS, best results in bold font.

County	Year	Num. Recs.	ABPM	EBPM	
				Euclid.	Manhat.
Aberdeenshire	2003	24000	0.9066	**0.8502**	0.8615
	2004	24000	0.9096	**0.8381**	0.8584
	2005	24000	0.9265	**0.8364**	0.8551
	2006	24000	0.9119	**0.8408**	0.8846
Cornwall	2003	24000	0.8666	**0.8112**	0.8499
	2004	24000	0.8972	**0.8322**	0.8611
	2005	24000	0.8668	**0.8244**	0.8345
	2006	24000	0.8995	**0.8382**	0.8675
Lancashire	2003	24000	0.8984	**0.8112**	0.8562
	2004	24000	0.9121	**0.8443**	0.8617
	2005	24000	0.8839	**0.8276**	0.8377
	2006	24000	0.9001	**0.8452**	0.8457
Norfolk	2003	24000	0.9016	**0.8463**	0.8685
	2004	24000	0.8991	**0.8529**	0.8740
	2005	24000	0.9166	**0.8267**	0.8605
	2006	24000	0.9151	**0.8533**	0.8786
Average		24000	0.9007	**0.8362**	0.8597

6.1 Comparison of BPM algorithms (ABPM and EBPM)

This section presents the results obtained with respect to the comparison of the BPM algorithms, founded on the banding score concept considered in this paper. The comparison was conducted by considering GBS and runtime. For the evaluation the operation of ABPM was compared with the operation of EBPM with either Euclidean or Manhattan distance calculation and with or without the use of M-Tables. The evaluation was conducted using the 16 5-D data sets extracted from the CTS database, each comprised of 24,000 records. The results with respect to the final GBS obtained are given in Table 1 and with respect to runtime (seconds) in Table 2. From Table 1 it can be observed that in all cases (as expected) the EBPM algorithm (both variations) produced better bandings than the ABPM algorithm. In addition Euclidean EBPM produced better bandings than Manhattan EBPM. The reason being that Euclidean distance measurement is more precise than Manhattan distance measurement and consequently Euclidean EBPM produced better bandings.

6.2 Comparison with Previous Work (BC and MBA)

This section reports on the experiments conducted to compare the operation of the proposed BPM algorithm with the previously proposed the BC algorithm [24] and the two variations of MBA algorithm, MBA$_{BFP}$ and MBA$_{FP}$ [21]. The comparison was conducted in terms of efficiency (runtime measured in seconds) and effectiveness. Although the BPM algorithms considered in this paper seek to minimise a GBS, the BC and MBA algorithms operated in a different manner. The BC algorithm sought to maximise a Mean Row Moment (MRM) value, while the MBA algorithms sought to maximise an accuracy value. An independent measure was thus used for the comparison. More

Table 2. Comparison of BPM algorithms in terms of runtime (seconds), best results in bold font.

County	ABPM	EBPM			
		With M-Table		Without M-Table	
		Euclid.	Manhat.	Euclid.	Manhat.
Aberdeenshire	**26.33**	36.85	30.27	75.41	63.61
	25.85	36.08	29.18	78.20	66.97
	22.35	34.42	28.42	75.31	60.93
	27.67	33.66	30.08	76.48	64.16
Cornwall	**25.20**	32.33	28.20	74.13	65.22
	26.75	38.86	30.84	73.13	63.38
	24.58	34.58	29.36	77.75	69.08
	23.49	36.38	30.31	72.20	67.45
Lancashire	**22.47**	39.21	28.01	79.59	63.03
	26.69	35.47	31.48	74.31	66.47
	25.45	32.25	28.94	69.16	59.34
	22.69	32.83	28.77	75.78	67.31
Norfolk	**26.91**	34.42	31.79	76.48	66.55
	22.91	35.70	27.72	73.88	62.52
	25.45	32.20	45.17	75.40	63.47
	24.99	33.07	29.78	69.22	61.55
Average	**24.99**	34.89	30.48	74.78	64.44

specifically the Average Band Width (ABW) measure was devised; the normalised average distance of dots from the diagonal measured according to the distances of the normals from the diagonal to each dot. ABW is calculated using Equation 5, where: D is the set of dots (with each dot defined in terms of a set of cartesian coordinates) and $maxABW$ is the maximum possible ABW value given a particular data matrix size.

$$ABW = \frac{\sum_{i=1}^{i=|D|} distance\ d_i\ from\ leading\ diagonal}{|D| \times maxABW} \tag{5}$$

For the evaluation the operation of only the EBPM algorithm was compared with BC, MBA$_{FP}$ and MBA$_{BFP}$ (because in 2-D both variations of the EBPM algorithm and ABPM operate in the same manner, it is only in higher dimensions that there operation differs). Note also that because BC, MBA$_{FP}$ and MBA$_{BFP}$ were only designed to operate in 2-D the comparison was conducted using 2-D data sets taken from the UCI Machine learning repository processed so as to produce binary valued equivalents. In each case the attributes represented the x-dimension and the records the y-dimension. The results obtained are presented in Tables 3 and 4. Table 3 shows the effectiveness results obtained in terms of GBS and the independent ABW measure. From the table it can be observed that the EBPM algorithm produce better bandings, in terms of GBS and the independent ABW measure, than the other banding algorithms considered (best result highlighted in bold font). From Table 4 it can also be observed that the BC, MBA$_{FP}$ and MBA$_{BFP}$ algorithms all require considerably more processing time than the proposed EBPM algorithm. For completeness Figure 2 shows the Lympography data set, before banding, and after banding using the EBPM algorithm.

Table 3. 2D banding evaluation results in terms of ABW and GBS, for the five banding mechanisms considered (best results in bold font)

Datasets	#	#	Before Banding		EBPM		MBA$_{FP}$		MBA$_{BFP}$		BC	
	Recs	Cols	ABW	GBS	ABW	GBS	ABW	GBS	ABW	GBS	ABW	GBS
Adult	48842	97	0.4487	0.3662	**0.3318**	**0.1294**	0.4116	0.2869	0.3617	0.2539	0.3394	0.4005
ChessKRv	28056	58	0.4444	0.3473	**0.2208**	**0.1791**	0.3816	0.3699	0.3246	0.2832	0.3240	0.3997
LetRecognition	20000	106	0.4125	0.3325	**0.2885**	**0.1682**	0.3407	0.2751	0.3152	0.2632	0.3246	0.3965
PenDigits	10992	89	0.4276	0.3453	**0.2197**	**0.2064**	0.3318	0.2775	0.2872	0.2874	0.3276	0.4005
Waveform	5000	101	0.4372	0.3402	**0.2414**	**0.2091**	0.3774	0.2958	0.2951	0.3215	0.2833	0.3702
Mushroom	8124	90	0.4297	0.3473	**0.2638**	**0.1774**	0.3866	0.3284	0.3845	0.3018	0.3297	0.3448
Annealing	898	73	0.4433	0.4133	**0.3630**	**0.1218**	0.4389	0.3300	0.3779	0.3162	0.3826	0.2977
HorseColic	368	85	0.4009	0.3857	**0.3205**	**0.2367**	0.4001	0.3801	0.3881	0.3760	0.3353	0.2904
Heart	303	52	0.4346	0.4318	**0.3016**	**0.1502**	0.4142	0.3387	0.3338	0.2833	0.3423	0.2651
Wine	178	68	0.4430	0.4564	**0.2027**	**0.2785**	0.3645	0.3970	0.3061	0.4015	0.3384	0.2561
Hepatitis	155	56	0.4438	0.4619	**0.2957**	**0.2063**	0.3032	0.4240	0.2962	0.4279	0.3438	0.2629
Lympography	148	59	0.3356	0.4581	**0.2826**	**0.2487**	0.2887	0.4359	0.2804	0.4540	0.3324	0.2738
Average	10255	78	0.4251	0.3905	**0.2777**	**0.1927**	0.3699	0.3449	0.3292	0.3308	0.3336	0.3299

Table 4. Run-time (RT) Results (seconds) Using UCI data sets.

Data sets	#	#	runtime (secs)			
	Rows	Cols	EBPM	BC	MBA$_{BFP}$	MBA$_{FP}$
Adult	48842	97	**76.74**	175.84	185.95	140.95
ChessKRvK	28056	58	**11.46**	23.27	27.90	27.81
LetRecognition	20000	106	**10.28**	26.38	24.54	21.31
PenDigits	10992	89	**02.81**	10.12	12.94	11.85
Waveform	5000	101	**0.88**	02.28	03.05	02.41
Mushroom	8124	90	**02.24**	08.47	09.07	08.14
Annealing	898	73	**0.05**	0.22	0.26	0.20
HorseColic	368	85	**0.02**	0.09	0.20	0.12
Heart	303	52	**0.02**	0.08	0.12	0.11
Wine	178	68	**0.01**	0.09	0.09	0.06
Hepatitis	155	56	**0.01**	0.08	0.06	0.06
Lympography	148	59	**0.01**	0.08	0.08	0.06
Average	10255	78	**08.71**	20.58	28.72	17.76

7 Conclusion

A number of BPM algorithms have been presented. More specifically two alternative banding algorithms were considered, ABPM and EBPM. Four variation of EBPM were considered, using Euclidean and Manhattan distance calculation, and with and without the use of M-Tables. The presented evaluation established the following main findings. The proposed BPM algorithms outperformed the previously proposed BC, MBA$_{FP}$ and MBA$_{BFP}$ algorithms, with respect to 2-D UCI data sets, in terms of: (i) efficiency, (ii) the GBSs obtained and (iii) an independent ABW measure. There is no difference in the operation of ABPM and EBPM in 2-D. In higher dimensions (5-D data sets were considered) the ABPM algorithm offered the advantage that it was more efficient than EBPM although the quality of the bandings produced were not as good as those produced using EBPM. In the context of EBPM the Euclidean variation produced the best quality bandings while the Manhattan variation was slightly faster. Use of M-Tables, to reduce the required amount of banding score calculation, was also found to be ben-

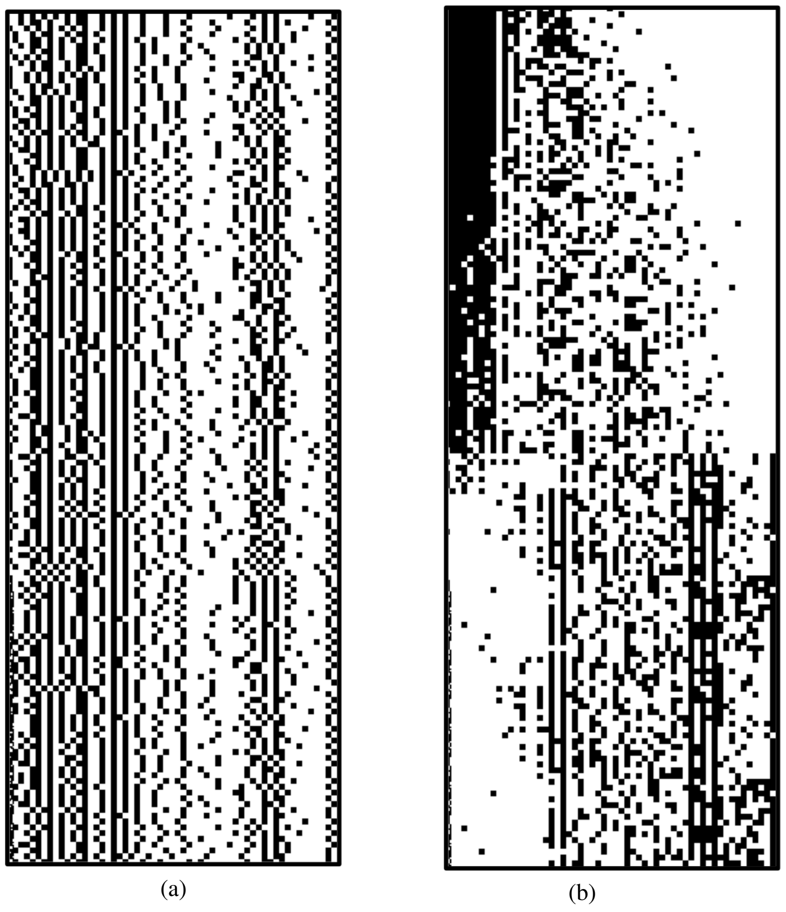

Fig. 2. Lympography rata set, before (a) and after (b) banding using EBPM

eficial. For future work the authors intend to investigate multi-core variations of the algorithms that can be used with respect to platforms such as the Hadoop distributed file store and data processing platform.

References

1. Fatimah B. Abdullahi, F. Coenen, and R. Martin. A novel approach for identifying banded patterns in zero-one data using column and row banding scores. In *International Workshop on Machine Learning and Data Mining in Pattern Recognition*, pages 58–72. Springer, 2014.
2. Fatimah B. Abdullahi, Frans Coenen, and Russell Martin. A scalable algorithm for banded pattern mining in multi-dimensional zero-one data. In *International Conference on Data Warehousing and Knowledge Discovery*, pages 345–356. Springer, 2014.
3. Fatimah B. Abdullahi, Frans Coenen, and Russell Martin. Finding banded patterns in big data using sampling. In *2015 IEEE International Conference on Big Data (Big Data)*, pages 2233–2242. IEEE, 2015.

4. Fatimah B. Abdullahi, Frans Coenen, and Russell Martin. Finding banded patterns in data: The banded pattern mining algorithm. In *International Conference on Big Data Analytics and Knowledge Discovery*, pages 95–107. Springer, 2015.
5. Fatimah B Abdullahi, Frans Coenen, and Russell Martin. Banded pattern mining algorithms in multi-dimensional zero-one data. In *Transactions on Large-Scale Data-and Knowledge-Centered Systems XXVI*, pages 1–31. Springer, 2016.
6. Farid Alizadeh, Richard M. Karp, Lee Aaron Newberg, and Deborah K. Weisser. Physical mapping of chromosomes: A combinatorial problem in molecular biology. *Algorithmica*, 13(1-2):52–76, 1995.
7. Jonathan E. Atkins, Erik G. Boman, and Bruce Hendrickson. Spectral algorithm for seriation and the consecutive ones problem. *Journal of Computing SIAM*, 28:297–310, 1998.
8. Kendall E. Atkinson. *An introduction to Numerical Analysis*. John Wiley & Sons, 2008.
9. C. Aykanat, A. Pinar, and U. Catalyurek. Permuting sparse rectangular matrices into block-diagonal form. *SIAM Journal on Scientific Computing*, 25:1860–1879, 2004.
10. R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley, 1999.
11. Jacques Bertin. *Graphics and Graphic Information Processing*. Walter de Gruyter, 1981.
12. Jacques Bertin. Graphics and graphic information processing. In *Readings in Information Visualization*, pages 62–65. Morgan Kaufmann Publishers Inc., 1999.
13. C. I. Blake and C. J Merz. Uci repository of machine learning databases. `http://www.ics.uci.edu/$\sim$mlearn/MLRepository.htm`, 1998.
14. Kuo-Young Cheng. Minimizing the bandwidth of sparse symmetric matrices. *Computing*, 11(2):103–110, 1973.
15. Kuo-Young Cheng. Note on minimizing the bandwidth of sparse, symmetric matrices. *Computing*, 11(1):27–30, 1973.
16. Yizong Cheng and George M Church. Biclustering of expression data. In *Ismb*, volume 8, pages 93–103, 2000.
17. Stephen B. Deutsch and John J. Martin. An ordering algorithm for analysis of data arrays. *Operations Research*, 19(6):1350–1362, 1971.
18. M. Fortelius, M. F./ Kai Puolamaki, and H. Mannila. Seriation in paleontological data using markov chain monte carlo methods. *PLoS Computational Biology*, page 2, 2006.
19. G. C. Garriga, E. Junttila, and H. Mannila. Banded structures in binary matrices. In *Proceedings Knowledge Discovery in Data Mining (KDD08)*, pages 292–300, 2008.
20. Gemma C. Garriga, Esa Junttila, and Heikki Mannila. Banded structure in binary matrices. *Knowledge and Information Systems*, 28(1):197–226, 2011.
21. E. Junttila. *Pattern in Permuted Binary Matrices*. PhD thesis, 2011.
22. Manfred Koebe and Jens Knöchel. On the block alignment problem. *Elektronische Informationsverarbeitung und Kybernetik*, 26(7):377–387, 1990.
23. Innar Liiv. Seriation and matrix reordering methods: An historical overview. *Statistical Analysis and Data Mining*, 3(2):70–91, 2010.
24. E. Makinen and H. Siirtola. The barycenter heuristic and the reorderable matrix. *Informatica*, 29:357–363, 2005.
25. Erkki Mäkinen and Harri Siirtola. Reordering the reorderable matrix as an algorithmic problem. In *International Conference on Theory and Application of Diagrams*, pages 453–468. Springer, 2000.
26. Heikki Mannila and Evimaria Terzi. Nestedness and segmented nestedness. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 480–489. ACM, 2007.
27. S. Myllykangas, J. Himberg, T. Bohling, B. Nagy, J. Hollman, and S. Knuutila. Dna copy number amplification profiling of human neoplasms. *Oncogene*, pages 7324–7332, 2006.
28. K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchical system structures. *IEEE Transactions on Systems, Man and Cybernetics*, 11:109–125, 1981.