Multimed Tools Appl
DOI 10.1007/s11042-010-0483-5

An incremental space to visualize dynamic data sets

Roberto Dantas de Pinho · Maria Cristina Ferreira de Oliveira · Alneu de Andrade Lopes

Abstract In Information Visualization, adding and removing data elements can strongly impact the underlying visual space. We have developed an inherently incremental technique (*incBoard*) that maintains a coherent disposition of elements from a dynamic multidimensional data set on a 2D grid as the set changes. Here, we introduce a novel layout that uses pairwise similarity from grid neighbors, as defined in *incBoard*, to reposition elements on the visual space, free from constraints imposed by the grid. The board continues to be updated and can be displayed alongside the new space. As similar items are placed together, while dissimilar neighbors are moved apart, it supports users in the identification of clusters and subsets of related elements. Densely populated areas identified in the *incSpace* can be efficiently explored with the corresponding *incBoard* visualization, which is not susceptible to occlusion. The solution remains inherently incremental and maintains a coherent disposition of elements, even for fully renewed sets. The algorithm considers relative positions for the initial placement of elements, and raw dissimilarity to fine tune the visualization. It has low computational cost, with complexity depending only on the size of the currently viewed subset, V. Thus, a data set of size N can be sequentially displayed in $O(N)$ time, reaching $O(N^2)$ only if the complete set is simultaneously displayed.

Keywords Dynamic data set visualization · High-dimensional data visualization · Multidimensional scaling · Projection

R. D. de Pinho (✉) · M. C. F. de Oliveira · A. de Andrade Lopes
Universidade de São Paulo, Caixa Postal 668, São Carlos, 13560-970, Brazil
e-mail: robertopinho@acm.org

M. C. F. de Oliveira
e-mail: cristina@icmc.usp.br

A. de Andrade Lopes
e-mail: alneu@icmc.usp.br

Published online: 29 January 2010

1 Introduction

High-dimensional data sets may be displayed by laying out their elements on a 2D space according to the similarity among items. This approach has been adopted to explore document collections [11, 30, 37], to display thumbnail collections of images [6, 34, 35] and to visualize general multidimensional data stored in databases [4].

However, to visualize a dynamic set where objects are constantly being added and removed, one can either (i) rely on a predefined layout, which would not capture major changes in data, or alternatively (ii) redo the whole map at critical steps, which might be disturbing to users as new map layouts may bear little or no resemblance to the original layout. We introduced a third strategy that incrementally positions and re-positions data items on a chess board-like space as elements are added and removed while preserving the overall layout configuration [33].

If we were to add chess pieces to a fixed position on a chess board, keeping only one piece by cell and similar pieces together, the arrival of a new piece would cause one or more pieces to be displaced to accommodate the new one. We could steer the way we displace pieces in order to achieve those goals. The problem could also be seen as one of sorting elements by similarity, but handling two dimensions, resembling somehow a 2D expanded insertion sort. When a new item arrives in an insertion sort, it may displace neighboring items, whilst the size of the allocated space increases.

Such a technique might be applied, e.g., to track changes in scientific literature. It would enable users to add new articles to maps that they are familiar with, maps that could also have been fine tuned by removing undesired papers. When new articles arrive, their location and impact on the map could provide these experienced users direct clues about their content and trends in research. This would not be possible if the choice is to redo the map or stick to earlier layouts. In any of those scenarios, the arrival of new elements can impact data organization on the underlying visual space. For instance, as a new research topic becomes popular, it could fill the map, squeezing together some loosely related articles. Consequently, the meaning of neither absolute positions nor distances is consistent across different time moments, a fact that can be very misleading to a human analyst. On following these examples and analogies, we can not rely on fixed visual axes, nor in absolute distances among elements. The important factor should rather be where elements are placed in relation to one another.

Based on the above rationale, we consider relative similarity of elements in the original high-dimensional space to place (project) them on a 2D space that follows a chess board analogy. The incremental strategy allows the effective display of dynamic data sets. The resulting visual space maintains a coherent disposition of elements from constantly updated sets, even though it does not rely on fixed, well defined or explicit visual axes or dimensions. Moreover, even if no item from the final set has co-existed with those from the initial set, their placement still follows the same global relative disposition of classes or categories as in the initial set of a given collection. Additionally, adding a new element does not demand a complete re-arrangement of elements. This is an interesting property, as users may track layout changes as element positioning gradually progresses. Moreover, the *incBoard* visualization does not suffer from element occlusion, and allocating a fixed screen space to each cell enables using sophisticated and highly informative glyphs.

We compared our approach with other multidimensional scaling strategies using objective measures and obtained very competitive results even when working with static data sets. To develop a proof-of-concept, the approach has been tailored to provide a specific visualization that handles ever changing corpora, one from which documents may be added, removed or replaced, which is not supported by current placement techniques. We provide a image thumbnail collection visualization example that benefits from the lack of occlusion on the board-like space, where a single element may occupy each cell. On a previous study, we have also found some evidence that using space-filling glyphs instead of point-like representations is beneficial to users [32].

Despite its advantages, *incBoard* has the shortcoming of not conveying pairwise similarity among neighboring elements. Other visualization systems typically convey pairwise similarity by the distance between two elements [4, 11, 37]. However, as *incBoard* fits elements in a grid of fixed size cells, neighbors are always placed at equal distances from each other, although neighboring elements are not necessarily equally similar.

We hypothesize that a more precise representation of pairwise similarity can better support investigation and exploration, as users might clearly identify and “value” neighbor elements relationships, and also identify clusters of similar elements. *HexBoard* has been proposed as an improvement to *incBoard*, capable of conveying similarity information for each neighboring pair [31]. It uses hexagon shaped cells, thus all neighboring pairs share an edge, whose visual properties are manipulated to reflect similarity between the neighboring elements. Nonetheless, neighboring elements are still placed at fixed distances from one another.

In this paper we introduce an alternative strategy that employs pairwise similarity from grid neighbors, as defined in *incBoard*, to freely reposition elements on the 2D space according to a layout solution provided by the incremental board, thus building an incremental visual space (*incSpace*) free from constraints imposed by the grid. Each element in the new incremental space has a corresponding position on the board, which continues to be updated and may be displayed alongside the new space. The new layout solution resembles the layout typically provided by other multidimensional scaling strategies, where similar elements are clustered together, while still being able to efficiently display dynamic data sets.

This paper presents the incremental space and revisits the *incBoard* solution, providing an additional usage scenario and also describing its first extension, the HexBoard and its corresponding visualization. The following section briefly describes related techniques and potential applications. Section 3 (i) details the board analogy space, (ii) presents how distances may be computed over it, (iii) describes how elements are added, removed and replaced from the board, and (iv) discusses algorithm complexity. Section 4 highlights differences between *incBoard* and *HexBoard*. Section 5 describes how the incremental board space can be turned into an incremental space. Section 6 presents the *incBoard*, *HexBoard* and *incSpace* visualizations. Section 7 brings case studies introduced both to compare the proposed techniques with other layout solutions using quantitative measures and to explore possible application scenarios. Finally, conclusions and further work are discussed in Section 8.

2 Related work

Visualizing document collections is a typical application of high-dimensional data visualization. Document collection visualizations are implemented in systems such as CiteSpace II [11] and the Projection Explorer—PEx [30]. On those systems and also on other related knowledge domain visualizations [8, 10], typically no conceptual and/or explicit meaning is assigned to each visual dimension. The same can be said about the FastMapDB tool [4], which employs 2D and 3D layouts to display collections of structured data from databases. Notable exceptions are the explicit mapping of height in 2.5D representations, scatterplot-based views, and, on some systems, the mapping of time to one of the visual axes. In these latter examples, only a few of the original data dimensions are explicitly mapped to a visual dimension, not contemplating a true solution for n-dimensional data. Some systems, such as PNNL's IN-SPIRE [37], apply techniques that rely on extraction of factors, such as Principal Component Analysis (PCA) and Latent Semantic Analysis (LSA) to layout documents. They might be able to represent more than two or three original dimensions on a two-dimensional space, but mapping only the first few significant or "latent" dimensions, respectively. Hence they are still inadequate to handle high dimensional data [29].

Nonetheless, in any of those solutions, visualizing changes in data requires building new layouts, or, in some cases, adopting visual dimensions established a priori from an initial set [38], which may be inadequate to layout latter versions of the data set.

One solution to this problem, as employed by Chen in his Citespace II visualization [11], is to build the map beforehand from the whole data set and then manipulate some visual attribute (for example, transparency) to add or remove elements from the representation. Although this solution gradually presents the evolution of a data set while maintaining a consistent layout, it cannot be considered as truly dynamic, since the complete collection must be fully available before the visual representation can be constructed. The map is not incrementally built as new documents are added to the corpus, removed from it, or yet when existing documents are replaced with new ones.

The *incBoard* grid-based space immediately reminds of self organizing maps (SOMs) for visualizing document collections [16]. However, those results bear major distinctions to ours: (i) grid dimensions are fixed beforehand, (ii) a single cell is allowed to hold multiple documents or elements, and (iii) once elements share a cell, similarity relations can only be inferred for the cell, relations among individual elements are not represented. Similarly to our approach, SOM's algorithm complexity is a function of the number V of map units (grid cells), not of the data set size N [15]. Nevertheless, to display a large data set using a small grid size V, a SOM would fit all elements into the available cells, whereas our approach displays them sequentially, having V elements on display at any given time, which is more suitable for dynamic and time stamped data.

Dynamic extensions to SOM have been proposed, such as Incremental Grid Growing(IGG) [7], and the growing self-organizing map (GSOM) and subsequent developments [1, 2]. They overcome the requirement for pre-defining grid dimensions, nonetheless many data items may still cluster together in a single cell (neuron), while other cells remain empty. This feature is useful to provide high level visualizations or to build hierarchical solutions, at the expense of not presenting individual elements. An undesired feature of IGG and GSOM, also pointed out elsewhere [27], is that they grow only from the border, leaving high density areas (many elements in each cell) at the center. Results from IGG seem to degrade when growing branches or arms merge. Latter versions of GSOM try to tackle this problem, but require smoothing phases that increase the complexity of the solution. Description of a removal process for neither GSOM or IGG has been found hitherto.

Chalmers briefly mentions that new elements might be added while his multidimensional scaling (MDS) technique converges to a solution, though this possibility is not further explored [9]. Once the projected data set becomes stable, he expects that reaching a solution will require a few $O(N)$ iterations of a stochastic force directed placement procedure. Latter enhancements to the technique also require a fine-tuning phase to reach the final solution [26]. In either case, complexity would lie well over $O(N^2)$ in an incremental scenario.

A more robust solution is presented by Law & Jain for an incremental ISOMAP algorithm [20]. However, coordinates for all data items must be updated and can change dramatically after each element addition [21]. Besides requiring extra computation to update the whole set, a global rearrangement of coordinates is not desirable for our goals. In contrast, our solution requires few changes on coordinates for each added (or removed) element, as discussed in Section 8.

Basalaj [5] introduces an incremental multidimensional scaling procedure. However, his approach requires a previously computed minimal spanning tree for the data set, thus undermining its application on dynamic data sets. Like ourselves, he advocates grid-like visualizations of elements arranged by similarity, the so-called Proximity Grids. He introduces some options on how to build a grid from a final MDS layout. Our approach is the exact opposite: we start with a grid layout and then build a MDS layout. Basalaj's *bump* approach is somewhat similar to our element addition procedure, although it does not consider the re-arrangement of the existing elements to reflect the changes introduced by the arriving ones. It is thus not appropriate for incremental building, being dependent on a previously built layout.

Related studies by Rodden et al. [34, 35] show some evidence that arranging images by similarity is useful to designers searching for photographs and also that users desire to avoid overlap. The Photomesa application [6] applies these principles to display clusters and hierarchically organized image collections employing treemaps and a simpler layout for image clusters called bubblemaps. The *incBoard* technique accomplishes both goals of layout by similarity and occlusion avoidance, with the additional benefit of gradually updating the visualization. A possible application of our solution is thus to maintain a personal image library that gradually receives additional images and yet keeps a familiar layout to the user.

Today, tag clouds are a popular feature of web pages, news sites and blogs. These are lists of frequent keywords selected by users to label ('tag') content, usually presented in alphabetical order, or ranked by frequency. Their spatial layout according to some similarity measure has been suggested before. Hassan-Montero and Herrero-Solana [13], for example, cluster them and display each cluster in separate lines. Lines are then sorted to approximate related clusters. As our grid space could be easily rendered using common HTML and no occlusion ever occurs, presenting tag clouds arranged by similarity is another possible application.

3 The Incremental Board space

The Incremental Board (*incBoard*) space is a grid-based 2D space with cells arranged in rows and columns. It is expected that only a single element should occupy a cell at any given moment, except temporarily when adding items. Formally:

An element E_i placed on this board space is represented by a point p_i with coordinates (x_i, y_i), with $x_i, y_i \in \mathbb{Z}$. In other words, $p_i \in \mathbb{Z}^2$.

The incremental board space takes two possible states: (i) a stable state, where no two elements share the same cell coordinates and (ii) an unstable state, where two or more elements share a single cell. The unstable state requires action in order to bring the board back to a stable condition.

The distance between two elements over the board is computed using the Chebyshev distance $d_c(E_i, E_j)$:

$$d_c(E_i, E_j) = max\left\{|x_i - x_j|, |y_i - y_j|\right\} \tag{1}$$

The Chebyshev or chessboard distance considers the number of steps required to move from one cell to another, as defined in the dynamic construction procedures detailed in Section 3.1. On the chess board analogy, it reflects the movement of the king.

An earlier attempt employed the Manhattan distance, which, on the chess board analogy, reflects the movement of the rook:

$$d_m(D_i, D_j) = |x_i - x_j| + |y_i - y_j| \tag{2}$$

Apart from closely sticking to the chess board analogy and its distance relations, an additional benefit of (1) is its more consistent behavior regarding the computation of distances from a cell to its neighbors. As shown in Fig. 1b, computing the Manhattan distance for the eight direct neighbors of a central cell produces different results for the four neighboring cells at the corners and the other four neighbors. Meanwhile, the Chebyshev distance is one for all eight direct neighbors (see Fig. 1a). After some early experiments, the Chebyshev distance was adopted in all examples presented here, as it yields a better use of the screen space. The Manhattan distance often results in a cross-shaped distribution of documents, with more documents placed around the axes.

The placement of elements on the 2D board may reflect their relative positioning on the original n-dimensional space or their relative ranking derived from some

2	2	2	2	2
2	1	1	1	2
2	1	0	1	2
2	1	1	1	2
2	2	2	2	2

(a)

4	3	2	3	4
3	2	1	2	3
2	1	0	1	2
3	2	1	2	3
4	3	2	3	4

(b)

Fig. 1 Comparison of neighbor distances using Chebyshev (**a**) and Manhattan distances (**b**), (1) and (2) respectively

Table 1 Relative position ranking example

E_j	$d_c(E_1, E_j)$	$R_{c1}(E_j)$	$\delta(E_1, E_j)$	$R_{n1}(E_j)$
E2	5	2	1,000	2
E3	4	1	850	1
E4	6	3	5,000	3

$R_{c1}(E_j)$ is the ranking on the 2D space relative to E_1. $R_{n1}(E_j)$ is the ranking on the nD space relative to E_1

dissimilarity measure evaluated between elements. So, for each element E_i on the board an error can be calculated to reflect the difference in the ranking of other elements when ordering them by their distances on the 2D space $d_c(E_i, E_j)$ as compared to their ranking by dissimilarities $\delta(E_i, E_j)$, defined in the original n-dimensional or conceptual space. On the hypothetical example presented in Table 1, the error should be zero, as the sorting of other elements relative to E_1 would result in the same ordered set $\{E_3, E_2, E_4\}$, using any of the distances and ranks shown.

We weight the error measure both by: (i) the difference from the expected position ($R_n i$) to the actual position ($R_c i$) of each of the other elements, and (ii) the expected position ($R_n i$), assigning more weight to errors originating on the vicinity of the reference element E_i.

For a list L of elements, the weighted error $W_{err}(E_i, L)$ relative to an element E_i is given by:

$$\sum_{E_j \in L} |R_{ci}(E_j) - R_{ni}(E_j)| \times (|L| - R_{ni}(E_j)) \tag{3}$$

Early experiments showed that the above error measure often yields the same value for different placement options (see Section 3.1). If this is the case, a weighted error count $C_{err}(E_i, L)$ is adopted as a second criterion:

$$\sum_{E_j \in L} \begin{cases} |L| - R_{ni}(E_j) & \text{if } |R_{ci}(E_j) - R_{ni}(E_j)| \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

Having defined these error measures, we can pursue the "goal of obtaining a monotone relationship between the experimental dissimilarities or similarities and the distances in the configuration[projection]", as stated by Kruskal [19] for nonmetric multidimensional scaling (MDS).

3.1 Dynamic construction

Except for the first element, new elements are always added, one at a time, at the cell that currently holds its most similar element. As the chosen cell becomes unstable, a process is started to bring the board back to a stable state. The first element is simply positioned at an arbitrary location.

The process of finding the most similar element can be costly, so the system operates in two possible modes: (i) full or (ii) stochastic sampling.

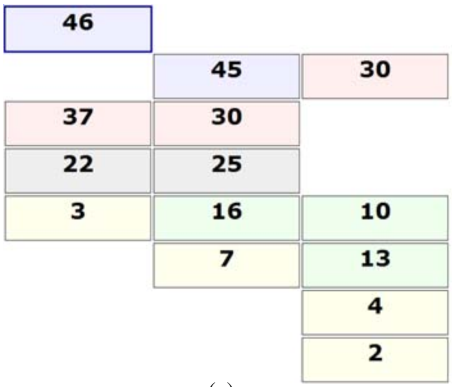
In full sampling mode, the arriving element is compared to every element on the board, whilst in the stochastic mode a fixed list of close neighbors is used instead. The stochastic sampling mode is derived from Chalmers et al. [9]. Besides the list of neighboring elements, a list of randomly selected elements is also kept. These two lists are used to update the weighted error measure when elements are moved during the addition or removal processes detailed below.

For a cell temporarily holding two elements, 16 solutions are evaluated to bring it back to a stable state. Eight of them keep the new element at the center cell and move the old one to one of its eight neighboring cells, and eight alternative solutions keep the old element at the center and move the new one. The one option that introduces the lowest added error for both elements is chosen.

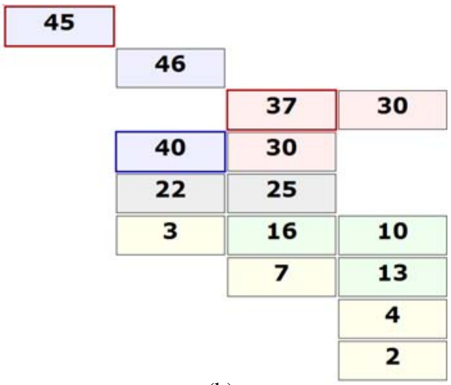
Eventually, the displaced element will fall on another already occupied cell, in which case the process is repeated with the two elements sharing this new cell. In order to avoid cycles, a list is kept of the already visited cells, which are not considered again as an option. The process ends when a moved element falls on an empty cell. The standard steps for adding an element are presented in Algorithms 1 through 5.

If all neighboring cells of an unstable cell have already been visited, the disturbing element is trapped and a special greedy procedure is applied to move it until it finds a non-trapped cell (procedure call at line 27 of Algorithm 1). At each step, the trapped cell's neighboring cells are evaluated and the one associated with the minimum error is chosen. Rows and columns left behind on the previous step are not considered any more. For example, if the element is moved from row 5 to row 6, only rows $[6, \infty[$ are considered in the future. The procedure stops when a cell with non-visited neighboring cells is found. Then, the regular process can resume. The list of visited cells is always cleared before adding a new element.

Figure 2 shows a board before and after adding the element *40*. The latest added element is shown with a blue border, while displaced elements are shown with a red border. On Fig. 2a, element *46* is initially added where element *22* is and then moved to the upper left corner of the board, displacing no element. When *40* is added, however, it causes the displacement of both *37* and *45*. Element *40* is first assigned to the cell where *22* is. The best option is evaluated and *40* is moved to where *37* was.



(a)



(b)

Fig. 2 Two moments of a board, before and after adding element *40*, respectively

Between moving *37* and *40*, *37* is chosen, and moved to where *45* was. This time, *45* is chosen and moved over *46*. *45* is chosen once again and moved to an empty cell, thus ending the process.

When removing elements, the board layout is adjusted pursuing two goals: (i) better reflecting relative positions once an element has been removed, and (ii) keeping the board as dense as possible, avoiding scattered elements. So, we try to move elements closer to the board's center. If the just emptied cell is above and to right of the center, we fill it with an element taken from one of the three cells above or to the right of it, choosing the option which yields the lowest added weighted error. The process is repeated with the newly emptied cell until all the three options are empty cells.

The most costly operation when adding or removing elements is to update the error measures for all the positioning alternatives considered. To handle this effectively, the process can select between operating in the full sampling mode or in the stochastic sampling mode.

The full sampling mode simply uses the current list of elements on the board B as the list L to compute weighted errors. This is a suitable option on maps with a low number of elements, or when changes in the data set are rare. Using the full sampling mode, we were able to obtain satisfactory performance when handling up to 400 elements simultaneously on the board, with performance degrading rapidly after that mark (see Section 3.2).

In the stochastic sampling mode, weighted error calculations for each element consider a close neighbors list and a random list of elements, both having fixed size [9]. Both lists start empty and are filled with randomly selected elements, with the ones most similar to the reference element kept in its close neighbors list. Whenever an element is a candidate for movement, its lists are updated, clearing and repopulating the random list. The neighbors list is updated if any of the newly selected random elements is closer to the reference element than those previously on its neighboring list. The list of close neighbors of a newly added element E_i, LN_{E_i}, is also improved using the list of close neighbors of its current closest neighbor E_j, LN_{E_j}. This improvement process is repeated if a different closest neighbor is found.

The operating mode may be switched in real-time. For dynamic data sets, the decision is based on the rate of new elements received, whereas for static data the choice considers the elapsed time required to add new elements. Ideally, one should adopt the full sampling mode to add the initial hundreds of items, and switch latter on to the stochastic sampling mode. This mode would then work over a more stable layout, where the random choice of a particular item does not significantly impact the layout, while keeping the computational cost low.

3.2 Algorithm complexity

All required operations consider only those elements currently on the board. Therefore, a whole data set with size N can be sequentially displayed with a constant cost for adding and removing elements, as long as the viewing window (board size, or number of occupied cells) has a fixed size V. The overall complexity would be $O(V \times N)$ or simply $O(N)$, as V remains constant.

A particular case, further analyzed here, occurs when the final board size V matches the data set size N, thus producing a map of the complete data set.

Using the stochastic sampling mode, each movement (selecting which of the two elements in a shared cell should move and where to) requires a constant effort k, proportional only to the size of the neighbors and random lists. For each added element, a certain number of movements is required until a moving element reaches an empty cell. An alternative view of the process is to imagine that an unstable cell state is moving towards an empty space. If the board is evenly occupied (close to the shape of a square), the maximum distance from the initial unstable cell to the border is $\sqrt{V}$, meaning that the process should stop in $\sqrt{V}/2$ movements. Under this behavior, the overall complexity would be $O(N \times \sqrt{V}/2 \times k)$ or $O(N^{3/2})$, when displaying the whole data set at once ($N = V$).

However, the space is not always evenly used and the movement of the virtual unstable cell is not constrained to go towards the nearest edge, being free to wander on the board as long as it does not move to any previously evaluated space. So theoretically, in the worst-case scenario, all cells could be visited every time, bringing the complexity to $O(N \times V \times k)$. Thus, $O(N^2)$ is the worst possible complexity, considering the particular case where the viewing window size equals the data set size.

Algorithm 1: Add a new element to the board.

```
   Input: New element $E_1$, Board $B$, Mode $m$
1  if $|B|=0$ then
2      $E_1.pos \leftarrow (0,0)$; $E_1.Rpos \leftarrow (0,0)$; $B \leftarrow E_1$; return;
3  $E_2 \leftarrow MostSimilar(E_1, B, m)$; // Algorithm 2
4  $E_1.pos = E_2.pos$; $B \leftarrow E_1$;
5  updateRposition($E_1$); // incSpace only, Section 5
6  $currentCell = E_2.pos$; $visitedCells \leftarrow \emptyset$;
7  while $|currentCell.elements| > 1$ do
8      minError = $\infty$ ;
9      $E_1 = currentCell.elements_1$; $E_2 = currentCell.elements_2$;
10     updateLists($E_1, B, m$); updateLists($E_2, B, m$); // Algorithm 5
11     foreach $cell | d_c(currentCell, cell) = 1$ and $cell \notin visitedCells$ do
12         $E_1.pos \leftarrow cell$; $visitedCells \leftarrow cell$;
13         solError = Error($E_1$, B, m)+ Error($E_2$, B, m); // Algorithm 4
14         if $solError < minError$ then
15             minError = solError;
16             $E_{toMove} \leftarrow E_1$;
17             $cell_{whereToMove} \leftarrow cell$;
18         $E_1.pos \leftarrow currentCell$;
19         $E_2.pos \leftarrow cell$;
20         solError = Error($E_1$, B, m)+ Error($E_2$, B, m); // Algorithm 4
21         if $solError < minError$ then
22             minError = solError;
23             $E_{toMove} \leftarrow E_2$;
24             $cell_{whereToMove} \leftarrow cell$;
25         $E_2.pos \leftarrow currentCell$;
26     if $E_{toMove} = null$ then
27         solveTrap(); // Section 3.1
28     $E_{toMove}.pos \leftarrow cell_{whereToMove}$;
29     updateRposition($E_{toMove}$); // incSpace only, Section 1.5
30     currentCell = $cell_{whereToMove}$;
```

Algorithm 2: Find the most similar element to a given element.

```
   Input: Element $E_j$, Board $B$, Mode $m$
 1 L=getL(Element $E_j$, Mode $m$); // Algorithm 3
 2 $minDis = \infty$;
 3 foreach $E_i \in L$ and $E_i \neq E_j$ do
 4     if $\delta(E_i, E_j) < minDis$ then
 5         $minDis = \delta(E_i, E_j)$; $mostSimilar = E_i$;
 6 return $mostSimilar$;
```

Algorithm 3: Get reference list L of elements.

```
   Input: Element $E_j$, Mode $m$
 1 if $m = full$ then
 2     $L \leftarrow B$;
 3 else
 4     $L \leftarrow LN_{E_j} \cup LR_{E_j}$;// stochastic sampling mode
 5 return $L$;
```

Algorithm 4: Compute error for a given element.

```
   Input: Element $E_j$, Board $B$, Mode $m$
 1 L=getL(Element $E_j$, Mode $m$); // Algorithm 3
 2 $W_{err} = \sum_{E_j \in L} |R_{ci}(E_j) - R_{ni}(E_j)| \times (|L| - R_{ni}(E_j))$;
   $C_{err} = \sum_{E_j \in L} \begin{cases} |L| - R_{ni}(E_j) & \text{if } |R_{ci}(E_j) - R_{ni}(E_j)| \neq 0 \\ 0 & \text{otherwise} \end{cases}$; return $W_{err}, C_{err}$;
```

Algorithm 5: Update random and close neighbors lists.

```
   Input: Element $E_j$, Board $B$, Mode $m$
   /* $LN_{E_j}$ is a sorted set, sorted on similarity to $E_j$ */
 1 if $m$=$full$ then
 2     return;
 3 $LR_{E_j} \leftarrow \emptyset$;
 4 while $|LN_{E_j}| < NeighborSetSize$ do
 5     $LN_{E_j} \leftarrow NewRandomElement$;
 6 while $|LR_{E_j}| < RandomSetSize$ do
 7     $NRE \leftarrow NewRandomElement$;
 8     if $\delta(E_j, NRE) < Max(\delta(E_j, E_i | E_i \in LN_{E_j}))$ then
 9         $LN_{E_j} \leftarrow NRE$;
10         remove last from $LN_{E_j}$;
11     else
12         $LR_{E_j} \leftarrow NRE$;
```

In the best case scenario, each new element would fall in a cell which has an empty cell as neighbor, so the addition process could stop in a single step. Thus, the lowest possible complexity would be $O(N \times 1 \times k)$ or $O(N)$.

Therefore, algorithm complexity, when $N = V$, should lie between these two limits: $\{O(N), O(N^2)\}$. An empirical analysis using a corpus of 675 scientific articles

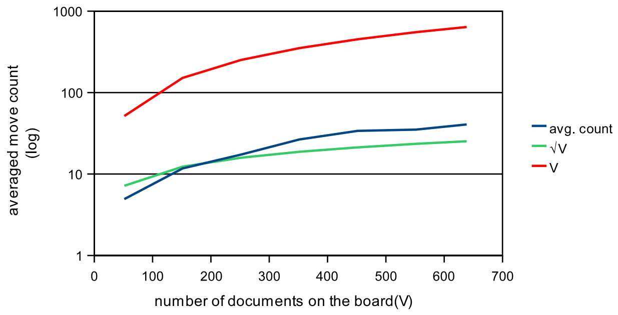


Fig. 3 Average number of movements required after adding a new document vs. number of documents on the board (V) when projecting the CIIS corpus

showed that the number of required movements stayed well below V for each element addition, close to the expected complexity of $O(N^{3/2})$, as shown in Fig. 3.

4 The HexBoard

The *HexBoard* [31] places elements on hexagon shaped cells instead of rectangular cells. By adopting hexagons, all neighboring pairs share an edge, thus it is possible to build a visualization that enhances *incBoard* in two ways: (i) it displays pairwise similarity or any other relevant pairwise information among neighboring elements; (ii) supports the identification of clusters and subsets of related elements by users.

The dynamic construction of the HexBoard follows the same steps as in *incBoard*. The major distinction lies on distance computation. Over the HexBoard, movements can follow three different axes, each of them perpendicular to a pair of edges. Thus, x and y coordinates are assigned to each cell, and an extra z coordinate is derived as $y - x$.

The distance can then be computed with a 3D version of the Chebyshev distance equation (1):

$$d_h(E_i, E_j) = max\left\{|x_i - x_j|, |y_i - y_j|, |z_i - z_j|\right\} \tag{5}$$

Algorithms 1 and 4 must be changed accordingly, at line 11 for Algorithm 1 and at line 2 for Algorithm 4.

Each cell has now only six neighbors: $\{(x-1, y-1, z), (x-1, y, z+1), (x, y-1, z-1), (x, y+1, z+1), (x+1, y, z-1), (x+1, y+1, z)\}$, which prompts changes on line 11 of Algorithm 1. Now, twelve solutions are evaluated when displacing elements, rather than sixteen, and there are only two options when removing elements, rather than three. These changes have no impact on algorithm complexity.

5 Breaking out of the board: the incremental space

The incremental space relies on an incremental board layout to provide the relative placement of elements. For every visualized element, two pairs of coordinates are kept: (i) a board position, defined in $\mathbb{Z}^2$, (ii) a "real" position, defined in $\mathbb{R}^2$, which we shall identify as its r-position.

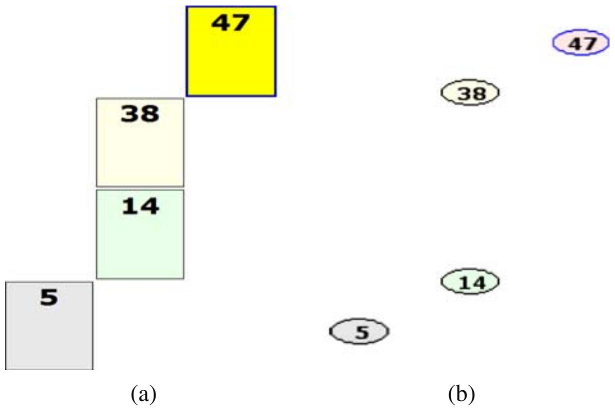


Fig. 4 Corresponding *incBoard* and *incSpace* visualizations

When an element is added to the board and whenever its board position changes, following the steps described in Section 3.1, its r-position is updated, considering dissimilarities to its surrounding neighbors on the board and their respective r-positions (see calls to *updateRposition* in Algorithm 1).

Figure 4 shows corresponding *incBoard* and *incSpace* visualizations. Dissimilarities were set as the differences between element numbers, i.e., $\delta(5, 14) = |5 - 14|$. On *incBoard* (Fig. 4a), distances between 5 and 14 and between 14 and 38 are exactly the same, one grid cell away. On *incSpace* (Fig. 4b), however, 14 is placed much farther apart from 38 than from 5. On the second image, an user without any knowledge of the content of these elements, could easily identify two clusters, {5, 14} and {38, 47}, whereas on the first image, the user might decide, from the visual clues, that there is a cluster with {5, 14, 38} and another one only with element 47.

The *updateRposition* procedure (not detailed as a separate algorithm) is described next.

First the centroid C of r-positions from the list of direct neighbors of an element $DN_{E_i} = \{E_j | d_c(E_i, E_j) = 1\}$ is computed:

$$C = \frac{1}{|DN_{E_i}|} \sum_{E_j \in DN_{E_i}} R(E_j) \tag{6}$$

where $R(E_j)$ is vector $[x, y]$ of r-coordinates of element E_j in $\mathbb{R}^2$.

The new or moved element is positioned at the centroid and then displaced based on the dissimilarities to its neighbors. The displacement Δ of element E_i is given by:

$$\Delta = \sum_{E_j \in DN_{E_i}} \left[\eta_j \times \beta \times \delta(E_i, E_j) \times \frac{\delta(E_i, E_j)}{\sum_{E_k \in DN_{E_i}} \delta(E_i, E_k)} \right] \tag{7}$$

where: (i) η_j can assume values $\{0, 1, -1, \sqrt{2}/2, -\sqrt{2}/2\}$ and depends on the relative positions of E_j and E_i on the board. If E_j is to the left of E_i, then η_j equals $[1, 0]$ when computing displacement for (x, y) coordinates in Δ, respectively, thus displacing E_i to the right; (ii) β is a constant that controls how spread the map is. It can be set by the user and accounts for differences in ranges for dissimilarities values;

(iii) the fraction that is part of the formula assigns greater weight to those neighbors that are the most dissimilar to the displaced element.

When an element E_i has only one neighbor E_j, its r-position will have a distance of $\beta \times \delta(E_i, E_j)$ to the r-position of E_j. If an element is found between two elements and is equally dissimilar to them, it will be placed in a position equally apart from them. If these two neighbors are at opposite positions (e.g. one over and one under E_i), their displacements will cancel out.

Building the incremental space has no impact on algorithm complexity, since computing the r-position of an element involves only its direct neighbors. There is no regularization or normalization steps, the only r-positions that change are those of the moved elements.

6 Visualizations

This section describes the peculiarities of *incBoard*, *HexBoard* and *incSpace* visualizations.

6.1 The incremental board visualization

The incremental board analogy enables a visualization without occlusion and with fixed screen space allocated to each data item, as illustrated in Section 7.2. Sophisticated glyph models can be designed to represent individual elements at multiple zoom levels. For fewer elements, the glyphs may carry the element's name, size or any other available information. They could, for example, display a miniature of a document's front page, or image thumbnails, as shown in Section 7.2. To display more elements, smaller rectangular glyphs may be chosen. Delimiting cells with grid lines is optional, as hiding the grid may result in a cleaner presentation and improved visualization.

Displaying newly added elements is an important component of the visualization, as is animation to help users tracking layout changes. Two viewing options are available: (i) a step mode, where the process is paused after each element addition, removal or replacement (a new element replaces an existing one), and (ii) a continuous mode, with no pause between operations.

A time or item count sliding window is available for viewing time sensitive data. So older elements are removed once their age is above a threshold or the item count grows above the limit. Glyph transparency may reflect element age, for instance, documents may be made to fade away prior to being actually removed from the visualization.

6.2 HexBoard visualization

On the *HexBoard* visualization, as neighboring hexagons always share an edge, visual attributes of each edge are manipulated to convey any relevant pairwise information available, as shown on Fig. 5. In that example, edge thickness and transparency have been adjusted to reflect neighbor similarity. Another option, not explored, but easily implemented is to map information to edge color.

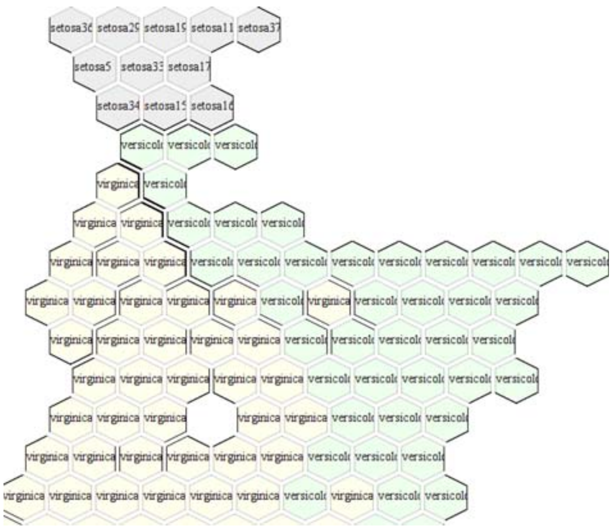

setosa36 setosa29 setosa19 setosa11 setosa37
setosa5 setosa33 setosa17
setosa34 setosa15 setosa16
versicol versicol versicol
virginica versicol
virginica virginica versicol versicol versicol
virginica virginica virginica versicol versicol versicol versicol versicol versicol versicol versicol versicol
virginica virginica virginica virginica virginica versicol virginica versicol versicol versicol versicol
virginica virginica virginica virginica virginica versicol versicol versicol versicol versicol
virginica virginica virginica virginica virginica versicol versicol versicol versicol versicol
virginica virginica virginica virginica virginica versicol versicol versicol
virginica virginica virginica virginica virginica virginica versicol versicol versicol
virginica virginica virginica virginica virginica virginica versicol virginica versicol versicol

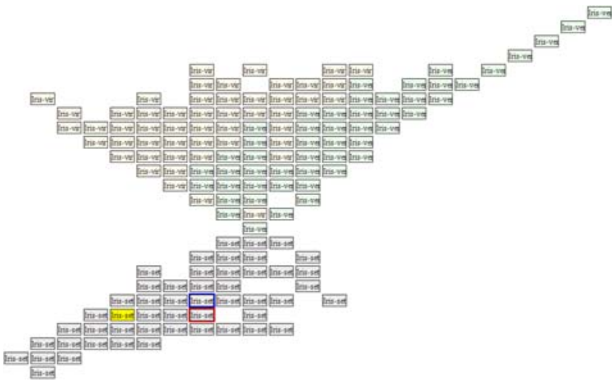
Fig. 5 Portion of a map of 150 samples from the Iris flower data set [3] belonging to three species: setosa (*grey*), versicolor (*green*) and virginica (*light yellow*). Thresholds were user selected

To adjust thickness and transparency, a γ value in the range [0, 1] is computed. Final thickness is then $\gamma\times$ maximum thickness, while transparency is given by $(1-\gamma)\times$ maximum α, where $\alpha = 0$ for transparent edges. To compute γ values, three options are available: (i) γ is simply set to match the dissimilarity for two neighbors $\delta(E_i, E_j)$, (ii) a user set threshold *th* can be chosen, having $\gamma = 1$, if $\delta(E_i, E_j) > th$, and having $\gamma \approx 0$ otherwise, or (iii) the value of γ is adjusted based on the median dissimilarity $\tilde{\delta}$, having $\gamma = \delta(E_i, E_j)/(2 \times \tilde{\delta})$, if $\gamma < \tilde{\delta}$, or $\gamma = \delta(E_i, E_j)/[(2 \times (max(\delta) - \tilde{\delta})]$, otherwise. For any of those options, the final result should present thicker and darker edges if two neighbors are not similar, and lighter, thinner ones if they are.

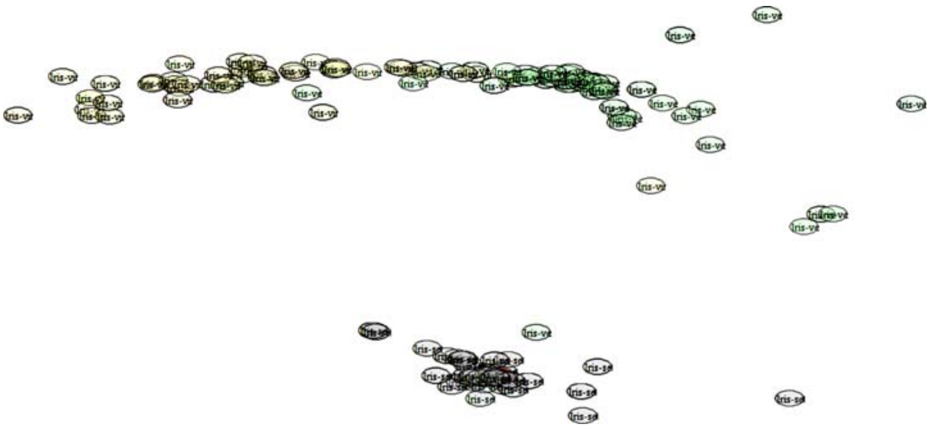
On the well known Iris flower data [3], containing of 150 data items belonging to one of three flower species, thicker borders (Fig. 5) separate the *setosa* from the *versicolor* species and partially separates the *versicolor* from the *virginica*. The map relies on a Euclidean distance calculation to evaluate flower similarity. Separation of species *versicolor* and *virginica* is not expected using this simple distance calculation.

6.3 Incremental space visualization

The incremental space visualization is akin to any visualization where elements are freely placed on a plane following some criteria, as in a scatterplot. Figure 6 shows very distinct views for the Iris flower data [3]. The figure underscores differences for these visualizations: (i) there is a better use of screen space on Fig. 5a; (ii) The setosa cluster is more clearly separated on Fig. 5b from the other two groups of flowers; (iii) Also on Fig. 5b it is possible to identify a few sub-clusters of elements, such as the cluster of five virginica flowers at the left; (iv) There is no occlusion on Fig. 5a. Despite these differences, it is noticeable that the general layout from Fig. 5a is preserved on Fig. 5b, where setosa flowers are found at the bottom of the map, virginica flowers mostly to the upper left and versicolor flowers mostly to the upper right.



(a) Incremental Board

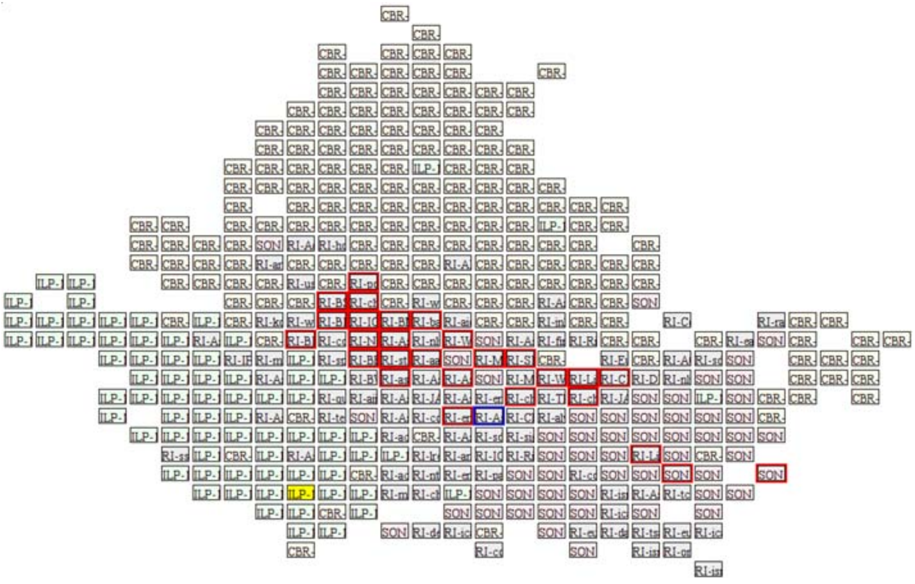


(b) Incremental Space

Fig. 6 Corresponding Incremental Board (**a**) and Space (**b**) visualizations of 150 samples from the Iris flower data set [3] belonging to three species: setosa (*grey*), versicolor (*green*) and virginica (*light yellow*)

This correspondence is the major particularity of the incremental space visualization and derives from the connection with the underlying approach to build and update the incremental space. To exploit this peculiarity, brushing was implemented in the proof-of-concept application, so the selection of a region on the incremental space visualization highlights the selected elements on the board. This feature is particularly useful to explore regions of the map where many elements are cluttered together. Since there is no occlusion in the incremental board, every selected element in the incremental space will be visible on the board.

The relation of the incremental board and the incremental space is more evident on a map of about 400 scientific articles, shown on Fig. 7. Due to the nature of the metric chosen (the cosine distance over a vector space model of the collection) and of the data itself, the incremental space layout more closely follows the grid


(a) Incremental Board

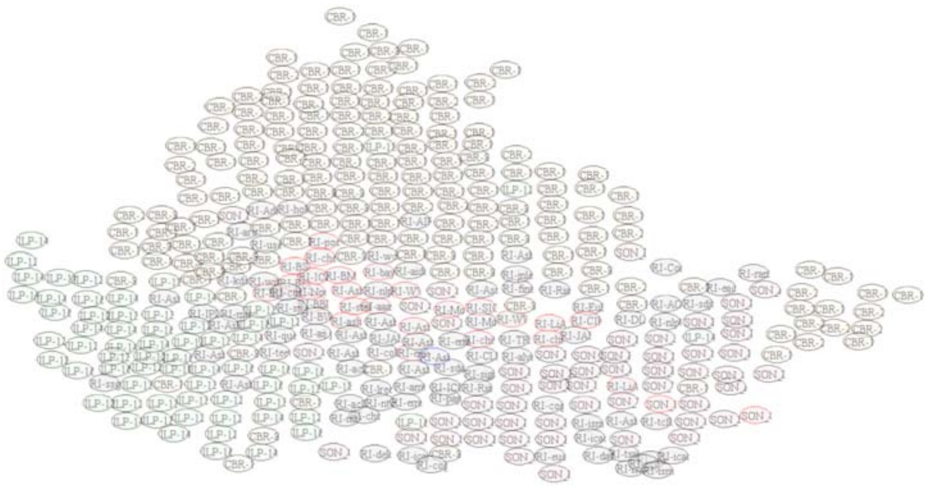

(b) Incremental Space

Fig. 7 Corresponding Incremental Board (**a**) and Space (**b**) visualizations of about 400 scientific articles. *Color* denotes manual classes assigned to each article

layout provided by the incremental board, as dissimilarity measures for neighboring elements do not vary as much as the Euclidean distance adopted with the Iris data set. However, it still fulfills its goals of clustering together groups of similar elements, such as the cluster of RI labeled articles (in gray) at the lower right or the somewhat detached cluster of CBR labeled articles at the far right of Fig. 7b.

A user may choose the incremental board as his or her main exploration window, using the incremental space as a guide to identifying regions of highly related elements.

7 Case studies

We initially present four case studies conducted to evaluate the incremental board space and the corresponding visualization. The first two studies compare *incBoard* with other MDS techniques using quantitative measures. The other two use the developed proof-of-concept application to explore potential usage scenarios of our technique: (i) displaying a collection of image thumbnails and (ii) exploring a collection of time-stamped news articles extracted from the Reuters Corpus [22].

Finally, we draw some comments comparing visualizations produced with *incBoard* and the corresponding ones built with *incSpace*.

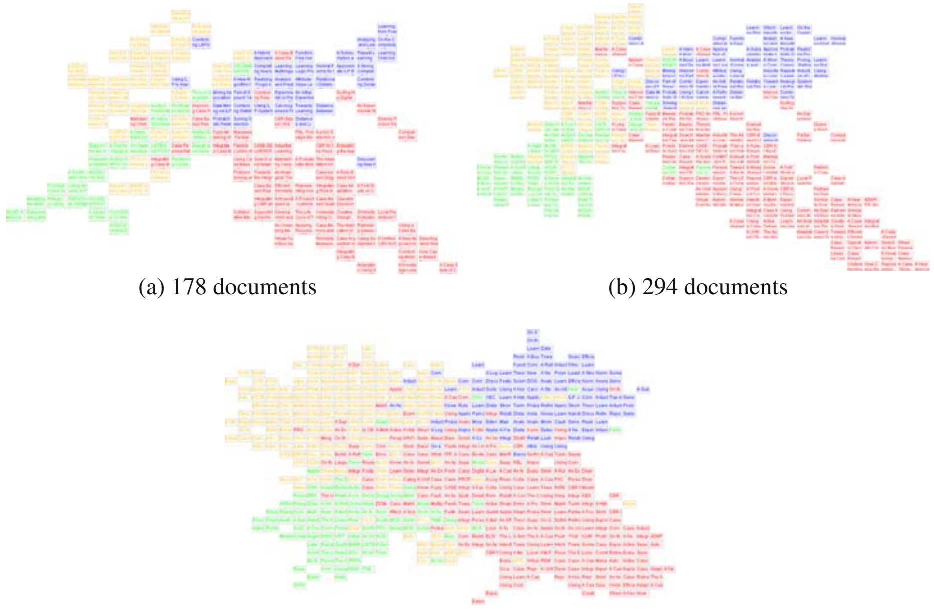
7.1 Comparing *incBoard* with MDS techniques

We employed stress [17–19] as a quantitative measure to evaluate *incBoard* as a MDS technique. In the original stress (8), $\hat{d}(E_i, E_j)$ is a fitted distance dependent on the dissimilarity $\delta(E_i, E_j)$. A simplified version of stress has been previously applied to evaluate projection techniques [9, 29] that assumes $\delta(E_i, E_j) = \hat{d}(E_i, E_j)$. As we favor relative positions, rather than trying to match expected distances (that is, we are dealing with a nonmetric MDS process), such an assumption does not hold and we must stick to the original hypothesis that $\hat{d}(E_i, E_j)$ is an unknown monotone distortion of $\delta(E_i, E_j)$ [19]. Moreover, distances in *incBoard* are in the range of $[1, \infty[$, while dissimilarities often fall in the range of $[0, 1]$, henceforth an exact match between $\delta(E_i, E_j)$ and $d_c(E_i, E_j)$ is not expected.

$$stress = \sqrt{\frac{\sum_{i<j}\left(d_c(E_i, E_j) - \hat{d}(E_i, E_j)\right)^2}{\sum_{i<j} d_c(E_i, E_j)^2}} \tag{8}$$

Assuming $\delta(E_i, E_j) = \hat{d}(E_i, E_j)$ also causes the stress measure to be sensitive to scaling: if we take a set of points A_1, simply scale their positions and call it A_2, stress will be found between the two sets. In visualization, scaling is an integral part of the process of displaying data on a given viewport. If we were to apply a "scale to fit" procedure, visualizing sets A_1 and A_2 would result in the exact same image, and yet they might have different stress measures when compared with a reference set.

As we understand that stress might not be a suitable metric to evaluate the placement strategy, due to reasons exposed by Paulovich et al. [29], we also evaluated the strategy with the Nearest Neighbors Precision metric (*nnp*), as defined by them. Given a labeled data set, it computes the percentage of neighbors that belong to the same class as the reference element. It may thus be used to verify whether the resulting *incBoard* layout is consistent with assigned labels or classes attributed to data items. We considered the first eight neighbors ($8 - nnp$), that correspond to the immediate cell neighbors on the board.



(a) 178 documents

(b) 294 documents

(c) 675 documents

Fig. 8 Three moments on the incremental construction of a visualization of the CIIS corpus. *Color* denotes manually assigned classes

We recorded measures for complete layouts and also mean values for measures computed as elements are gradually added in a random order. These latter measures reflect the technique's performance while the layout is incrementally built. Whenever a technique is subject to variance, mean average and standard deviation (std.dev.) measures were computed over 10 distinct runs. Projections compared with *incBoard* were built using the PEx tool [30] and its default settings.

The first data set used consists of 675 scientific articles manually classified into one of four subject areas: Case-Based Reasoning (CBR), Inductive Logic Programming (ILP), Information Retrieval (IR) and Sonification (SON)—this same dataset, henceforth the CIIS corpus, has been used before to compare several multidimensional projection techniques [29]. Document content dissimilarity was measured using the cosine distance over a vector space model of the collection. Stop words were removed and frequency-based Luhn's cuts applied to select relevant terms.

Figure 8 shows intermediate and final document maps of the corpus obtained with *incBoard*. Documents are colored according to their assigned classes. Notice that the general placement of classes remains globally stable as the visualization is constructed. Documents were added using the full sampling mode until the 300th element, from then on the insertion process switched over to the stochastic sampling mode. At that point, error calculations for each document used only the random and neighboring lists of elements. The size of the random and neighboring lists was set to 16 and 24 elements, respectively. The assigned classes played no role in the arrangement process, serving solely to select each element's display color.

Table 2 Results of different projection techniques for the CIIS corpus

Technique	$8 - nnp$ ($\times 10^2$)	Std.dev. ($\times 10^2$)	Stress ($\times 10^2$)	Std. dev. ($\times 10^2$)
PROJ	67.8	8.8	**17.6**	0.7
incBoard(final map)	67.8	4.3	18.2	0.6
PCA	**85.0**	–	21.6	–
LSP	70.7	5.9	22.3	2.1
Sammon's	76.0	–	37.7	–

Best results in bold

Table 2 shows stress and *nnp* results for the final layout and the compared techniques: (i) Projection by Clustering—PROJ [28], (ii) Least Square Projection—LSP [29], (iii) Principal Component Analysis—PCA [14], and (iv) Sammon's Mapping [36]. Stress used $\hat{d}(E_i, E_j)$ computed by the fitting algorithm proposed by Kruskal [17]. Results for both measures are very encouraging.

Table 3 shows results for stress and *nnp* computed while adding documents until reaching the total number of 675 documents. Their mean average is identified as *incremental*. Results are consistent throughout the process, indicating that views obtained at partial stages are also usable.

We noticed that maps with 200 elements or less consistently presented poorer results for the $8 - nnp$ measure, which could be evidence of a bias of the measure towards the size of the set (see Tables 3 and 4). For example, in a square with 16 elements belonging to two classes (8 elements on each), the lowest possible number of elements with a neighbor of a different class is 8 ($2 \times \sqrt{n}$), while in a square with 100 elements this number is only 20 ($2 \times \sqrt{n}$). In other words, the measure resembles the relation between perimeter and area of a shape, which is not constant for different areas. Based on this rationale, we should only compare *nnp* results on maps with roughly the same number of elements, and may also expect slightly lower results for the incremental averages in Tables 3 and 5.

The same corpus was also used in an example that resembles more closely the visualization of a dynamic corpus. In this second scenario, documents were incrementally added until 300 documents were placed on the board. After that, replacement takes in, with a document being removed whenever a new document is added, until all 675 documents in the corpus have been displayed. Figure 9 shows

Table 3 Stress & *nnp* measures computed while documents are gradually added to a board of 675 CIIS documents using *incBoard*

Documents on board	$8 - nnp$ ($\times 10^2$)	Std.dev. ($\times 10^2$)	Stress ($\times 10^2$)	Std.dev. ($\times 10^2$)
100	55.6	5.3	18.8	1.5
200	64.7	5.6	18.9	1.4
300	70.7	5.5	19.1	1.2
400	67.7	5.7	18.7	0.8
500	67.7	6.0	18.5	0.6
600	67.8	5.2	18.4	0.6
675	67.8	4.3	18.2	0.5
Incremental	66	7.1	18.7	1.0

Table 4 Stress & *nnp* measures computed while documents are gradually replaced from a board with 300 CIIS documents using *incBoard*

Documents added/ on board	$8 - nnp$ $(\times 10^2)$	Std. dev. $(\times 10^2)$	Stress $(\times 10^2)$	Std. dev. $(\times 10^2)$
100/100	56.92	4.56	18.56	1.09
200/200	65.58	3.12	19.11	1.19
300/300	69.86	2.71	19.44	1.35
500/300	64.42	4.65	18.91	0.66
675/300	68.45	4.84	18.51	0.71

Table 5 Results of different projection techniques for the Iris flower data set

Technique	$8 - nnp$ $(\times 10^2)$	Std. dev. $(\times 10^2)$	Stress $(\times 10^2)$	Std. dev. $(\times 10^2)$
PCA	**93.6**	–	**0.1**	–
LSP	91.8	2.6	3.0	1.1
PROJ	83.6	6.3	10.2	1.7
incBoard(final map)	86.4	2.5	14.3	2.5
incBoard(incremental)	81.9	5.4	14.5	2.6

Best results in bold

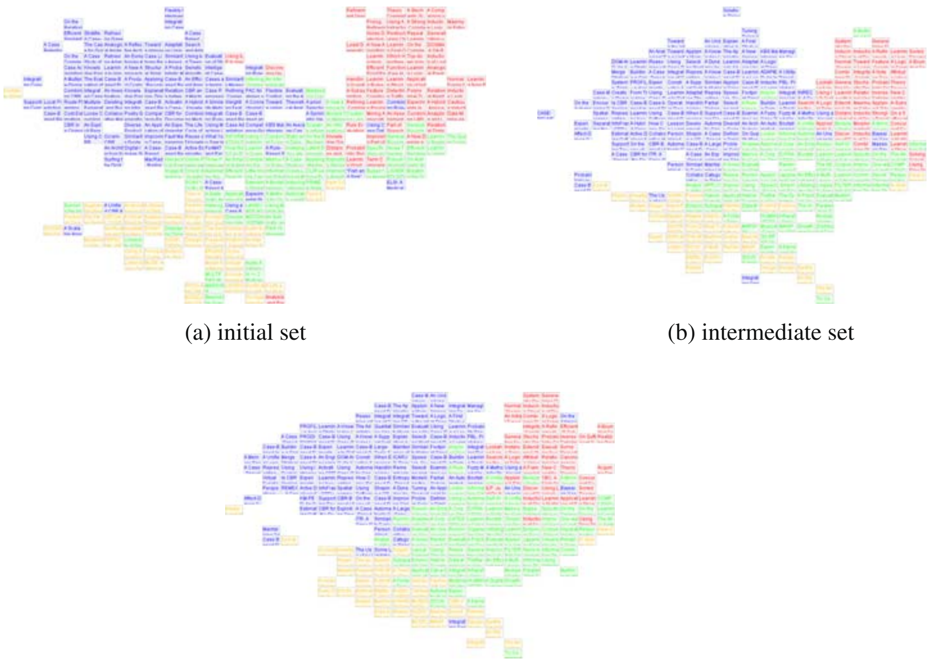


Fig. 9 Three moments on a visualization of the CIIS corpus where only 300 documents are kept at a time. *Color* denotes manually assigned classes

three distinct moments of the process. Indeed, inspecting the visualizations created along the process one observes that the general disposition of classes in the board is kept throughout.

Again, quantitative results are very consistent (see Table 4), highlighting the suitability of *incBoard* to display dynamic data sets.

The second case study was conducted on the already mentioned Iris flower data [3]. Our goal was to asses how our solution would perform when presented with a data set that is more easily presented in two dimensions. Euclidean distance was employed to compute similarity.

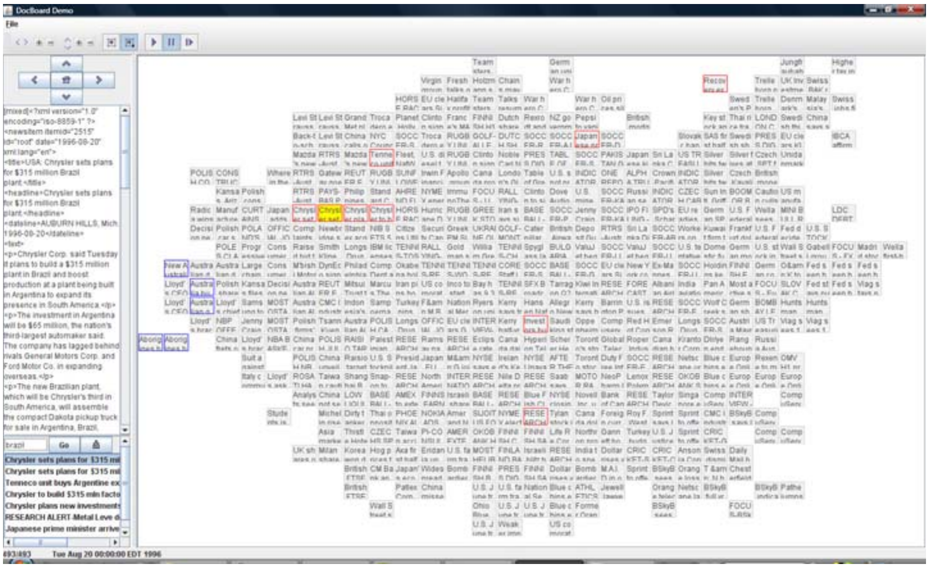
On this case study, stress results were fair (see Table 5), though not as good as in the previous one. From these results, one could argue that there exists a 2-dimensional solution for the projection of the Iris flower data set (stress for PCA is 0.1×10^{-2}). Therefore, the other techniques listed benefit from not being subject to the same constraints as *incBoard*, as they are free to position each data item at their ideal positions. Nonetheless, for more complex data sets, such ideal positions may not exist and, then, their advantage in this respect becomes irrelevant. Still, the *nnp* measure is again very close to the best results, possibly implying that even if an ideal positioning was not attainable, overall distribution of data items is good. Moreover, as it happened with the CIIS corpus, stress and *nnp* behaved consistently throughout the process (see incremental results in Table 5).

7.2 Potential usage scenarios for the incremental space

In the next case study, we tailored the visualization to sequentially display over 800,000 news articles from the Reuters corpus on a board with 500 cells. A dedicated thread sequentially extracts a batch of articles (a single day's worth of news) from the corpus and inserts them on a queue for use by the visualization system. Once that queue is empty, the thread processes another batch to replenish it. We used only the *headline* and *text* tags to build the vector space model. Although there are better solutions to compute a similarity measure for a dynamic corpus, we chose to simply take the first 50,000 documents to compute base term document frequencies. We acknowledge that in a real world scenario these initial documents would not be available, yet we believe that tackling this issue is beyond the scope of this paper.

A simple query tool is provided that computes the cosine distance between a query string and documents on the board and a list of hits is shown ranked by similarity(on the lower left panel in Fig. 10). Documents added to the board are also compared to the active query. The *incBoard* system has a document lock feature: when active, documents similar to the active query Q, $\delta(D_i, Q) < 1$, are not removed from the board, and other documents are chosen instead for replacement to maintain the desired number of documents on the board. We opted not to display any classification information, to keep the focus on the chosen features.

We began adding nearly 500 documents to the board, pausing, and then performing a search for 'Brazil', our home country. We found a few clustered articles concerning plans for a new Chrysler plant in Brazil (see Fig. 10). We then added 'Chrysler' to our active query. Documents having 'Brazil' and/or 'Chrysler' are shown with a red border. The currently selected document is shown with a yellow background color, and its source file is shown in the left upper panel.



```
[mixed]<?xml version="1.0"
encoding="iso-8859-1" ?>
<newsitem itemid="2515"
id="root" date="1996-08-20"
xml:lang="en">
<title>USA: Chrysler sets plans
for $315 million Brazil
plant.</title>
<headline>Chrysler sets plans
for $315 million Brazil
plant</headline>
<dateline>AUBURN HILLS, Mich
1996-08-20</dateline>
<text>
<p>Chrysler Corp. said Tuesday
it plans to build a $315 million
plant in Brazil and boost
production at a plant being built
in Argentina to expand its
presence in South America.</p>
<p>The investment in Argentina
will be $65 million, the nation's
third-largest automaker said.
The company has lagged behind
rivals General Motors Corp. and
Ford Motor Co. in expanding
overseas.</p>
<p>The new Brazilian plant,
which will be Chrysler's third in
South America, will assemble
the compact Dakota pickup truck
for sale in Argentina, Brazil,
```

```
Chrysler sets plans for $315 mil
Chrysler sets plans for $315 mil
Tenneco unit buys Argentine ex
Chrysler to build $315 mln facto
Chrysler plans new investment
RESEARCH ALERT-Metal Leve d
Japanese prime minister arrive
```

```
493/493    Tue Aug 20 00:00:00 EDT 1996
```

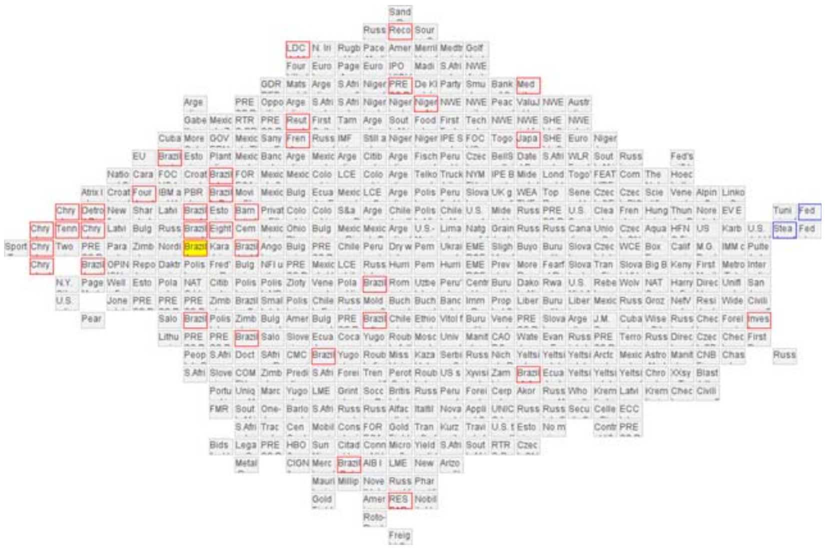
Fig. 10 The *incBoard* proof-of-concept application on document visualization. 493 documents from the Reuters Corpus shown

We activated the document lock feature and let new documents be added in continuous (‘play’) mode. When the number of documents added was approaching one thousand (and around 400 were removed to maintain the target board size), we noticed the appearance of a few more documents relevant to our query (see Fig. 11a). The new cluster included financial market news concerning Brazil. The ‘Chrysler’ cluster was still present, and unconnected with the financial cluster.

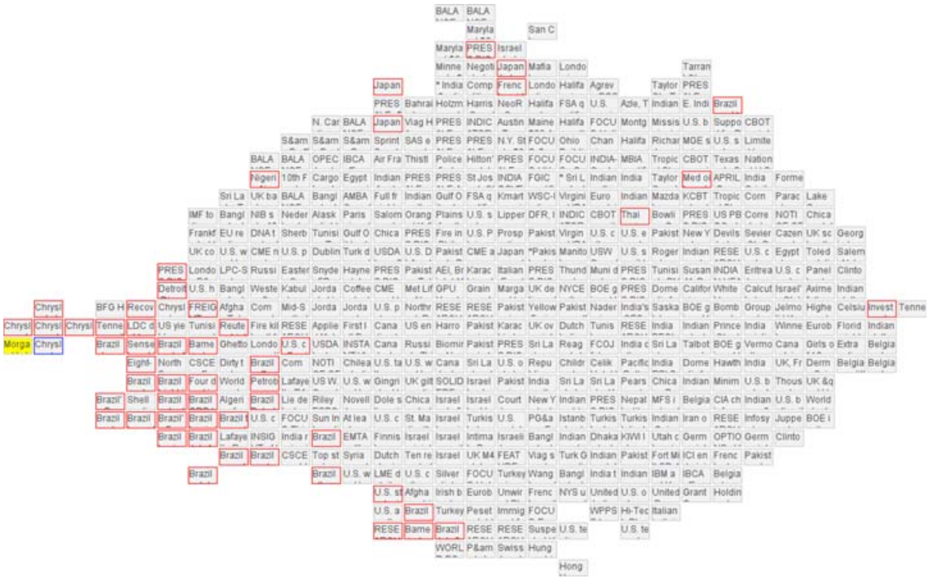
Latter on, after adding over 1,800 documents, documents similar to our query are found to be clustering together as they now represent a significant portion of the currently viewed set (see Fig. 11b). Yet, the Chrysler cluster is preserved (left of Fig. 11b). The selected document nearby (highlighted in yellow) addresses neither Chrysler nor Brazil, rather it relates to gas prices, thus being reasonable to find it placed close to articles on an automaker’s investment plans. On the other hand, query hits found farther away from the larger ‘Brazil’ cluster only briefly mention Brazil. For instance, the news article starting with ‘Nigeria’ (upper left on Fig. 11b) reports the closing of an air route between Nigeria and India by a Nigerian airline, and mentions that the same happened before with a route to Brazil.

The last case study for *incBoard* presents one extra application of our layout technique and visualization. It lacks formal user evaluation, that should be performed once a more concrete task or usage scenario is defined.

It illustrates the applicability of our layout technique to display a collection of images, benefiting from both a layout that can arrange images by their content and that does not suffer from occlusion (see Fig. 12). It uses a collection of 1,109 manually tagged images from [23]. These tags describe objects and features found on the images. Dissimilarity was computed employing the Jacquard coefficient to compare these manually assigned tag sets. Some clusters of images depicting the



(a) Nearly 1,000 documents added since the beginning of the visualization.



(b) *incBoard* Nearly 1,800 documents added since the beginning of the visualization.

Fig. 11 *incBoard* visualization of 500 documents from the Reuters Corpus, shown with 'Brazil Chrysler' as active query

same object or place can be noticed in Fig. 12. No image attribute was used in the comparison, and, yet, some areas where some colors are predominant can be found, perhaps reflecting some relation between tags and colors.

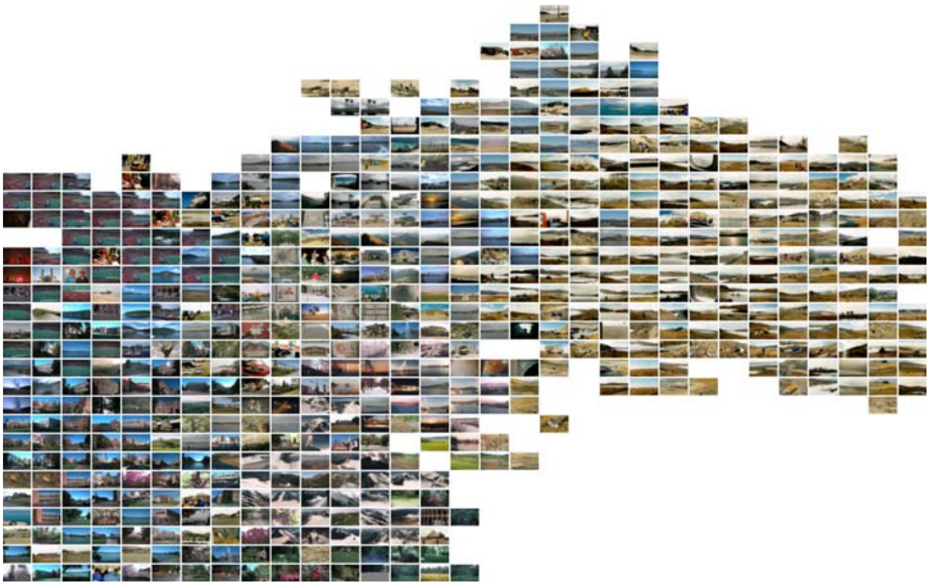


Fig. 12 Detail from a map of 1,109 manually tagged images. Similarity determined by tag comparison

7.3 Comparing *incSpace* and *incBoard*

Stress computed for the projection of the Iris data set was considerably lower than stress computed for the underlying incremental board. It decreased from an average of 12×10^{-2} (std.dev. 3×10^{-2}) to 6×10^{-2} (std.dev. 4×10^{-2}). The new value is still worse than the results for PCA (0.1×10^{-2}), but close to the results for LSP (3×10^{-2}) and better than PROJ (10×10^{-2}). These better results confirm our suspicion that since there exists a solution to layout this data set on the plane, the grid imposed an undesired constraint. Once that constraint is lifted, stress results approach those of other non-optimal techniques.

There is a slight improvement also for the $8 - nnp$ measure, from 88% to 90%. On the incremental board space, even if a perfect class separation is attained, there will be always elements which have as close neighbors elements from a different class. On the incremental space, however, since less similar neighbors are pushed away, it is possible that the closest eight neighbors of an element do not match exactly its eight neighbors on the board. Thus, if classes are clustered together, it is likely that those elements that have close neighbors of a different class on the board, may find their close neighbors on the incremental space amongst elements of the same class, thus improving the $8 - nnp$ measure.

Improvement in stress and $8 - nnp$ measures were much more modest for the CIIS corpus. Stress decreased from 18.3×10^{-2} (std.dev. 1.0×10^{-2}) to 18.1×10^{-2} (std.dev. 1.0×10^{-2}), while the $8 - nnp$ measure rose from 70.6×10^{-2} (std.dev. 6.0×10^{-2}) to 71.2×10^{-2} (std. dev. 5.0×10^{-2}). Measures for *incSpace* were always better than those for *incBoard*, except for one of the ten runs, for which stress was 0.23% worse. The layout provided by *incSpace* for this data generally follows the layout provided by *incBoard*, possibly as a result of the distribution of values of the dissimilarity metric chosen. The Euclidean distance metric adopted for the Iris data set has a broader range, thus yielding a map that is not as similar to the original board. Yet, results for the CIIS corpus with *incSpace* where at least as good as results for *incBoard*, while providing the additional benefit of clearly depicting clusters of similar items.

8 Conclusions and further work

We discuss layouts for visualizing multidimensional data objects based on an incremental projection strategy. On the core of our approaches is a placement strategy (*incBoard* [33]) that relies on the relative ranking of the displayed elements to position new elements on a partially filled board which resembles the way users might layout constantly arriving elements on a board, based on their content similarity. Absolute positions or distances bear little meaning on the resulting visual space, nonetheless, its layout follows relative ranking relations as found on the original (or conceptual) space.

The solution is particularly suitable to display dynamic data sets, as it is incrementally built and provides a built in mechanism for removing elements, while maintaining a compact layout and low computational cost. Moreover, adding a new element does not demand a complete re-arrangement of elements. An average of only 2.6% (std.dev. 0.1, computed over 10 runs) of existing elements were displaced after adding a new one using the CIIS corpus. This is an interesting property, as it allows users to track layout changes as element positioning gradually progresses.

Though the manually labeled document classes were not input into the layout process, classes were consistently kept in the same relative positions on the screen, during the gradual construction of the visualization of a whole corpus and also when elements of the corpus were gradually replaced. A consistent relative arrangement of classes was achieved even after a complete renewal of the viewed document set was forced, by replacing viewed documents until no document from the original viewed set remained. Although the *incBoard* approach assigns no explicit meaning to each visual dimension, this feature is particularly interesting, as it helps users to maintain a mental map of the arrangement.

The *incBoard* visualization does not suffer from occlusion of elements, and allocating a pre-defined screen space to each cell enables using sophisticated and highly informative glyphs. The user can also adjust, at any moment, the number of elements to be displayed simultaneously. The approach could be easily integrated into an operating system to provide file system navigation, while using its standard icons and file representations. A limitation, however, is that it does not favor identifying relative pairwise similarity of neighbors. Neighbors being always placed at the same distance from each other, although not all neighboring elements are equally similar.

We thus expanded the solution by building an incremental space where elements are freely placed on the plane, relying simultaneously on the incremental board placement solution and the dissimilarity of an element to its neighboring elements. The new approach places similar elements close together while distancing less similar ones, resembling typical MDS solutions, while still being inherently incremental. The corresponding visualization may be easily combined with the incremental board visualization as every element on the incremental space has a corresponding position on the underlying incremental board. Quantitative analysis of the new space showed that it performs at least as well as the incremental board, having performed substantially better under certain circumstances.

A user evaluation of the suitability of the layout technique for specific tasks would be desirable to compare the incremental board space solution with other layout techniques. There is also room for improving specific steps of the process. For instance, the choice of the closest neighbor could use a better search strategy. Likewise, alternatives to the error measure adopted and which options to evaluate when choosing an element to move could be subject to further investigation. Current choices are the result of some early ad-hoc experimentation and follow the underlying reasoning presented here.

The next step on system development is to enrich the map with concepts and/or topic representations. Topics may be derived automatically from document contents using dynamically extracted Locally Weighted Rules [24]. These topical markers, once placed on the board, could steer the placement of arriving elements. They could also be placed and positioned by users, who could then adjust the layout according to their specific interests and needs. Support to such operations is possible by adapting the incremental build operations. Another possibility is to allow direct manipulation of items on the board. The use of visual text mining tools such as the one taylored for the Reuters corpus case study (Section 7.2) was suggested to aid systematic reviews of scientific literature [25]. It is often the case where one needs to update a review or survey. In this scenario, an incremental visual text mining tool could be of great use, since newly found articles could be easily added to existing maps of the selected literature.

Adapting the technique to place elements of different shapes or sizes could allow its application to other problems. For instance, one could envision an automatic page layout algorithm that instead of placing elements according to a fixed order [12] would render a newspaper page where similar stories would be closer together. A whole newspaper could thus be automatically rendered, perhaps including only news stories that suit a user's needs, and yet looks like a manually produced one.

Acknowledgements The authors thank the financial support of FAPESP (Grants 2005/02263-2 and 2008/046228) and CNPq (Grant 305861/2006-9). Part of Roberto Pinho's research was conducted while visiting Drexel University under the supervision of Dr. Chaomei Chen and the support of a CAPES PDEE Grant (BEX 0651-07-9). He wishes to thank Dr. Chen and fellow researchers at Drexel University for their invaluable contributions.

References

1. Alahakoon D, Halgamuge S, Srinivasan B (2000) Dynamic self-organizing maps with controlled growth for knowledge discovery. IEEE Trans Neural Netw 11(3):601–614
2. Amarasiri R, Alahakoon D, Smith K, Premaratne M (2005) HDGSOMr: a high dimensional growing self-organizing map using randomness for efficient web and text mining. In: Proc. IEEE/WIC/ACM int. conf. on web intelligence, pp 215–221
3. Asuncion A, Newman D (2007) UCI machine learning repository. University of California, Irvine, School of Information and Computer Sciences. http://www.ics.uci.edu/~mlearn/MLRepository.html
4. Barioni M, Botelho E, Faloutsos C, Razente H, Traina, A, Júnior C (2002) Data visualization in RDBMS. In: Proc. IASTED int. conf. information systems and databases ISDB, pp 264–269
5. Basalaj W (2000) Proximity visualization of abstract data. Ph.D. thesis, University of Cambridge Computer Lab. http://www.pavis.org/essay/

6. Bederson BB (2001) Photomesa: a zoomable image browser using quantum treemaps and bubblemaps. In: UIST '01: Proc. of the 14th annual ACM symp. on user interface soft. and technology, New York, NY, USA, pp 71–80. doi:10.1145/502348.502359
7. Blackmore J, Miikkulainen R (1995) Visualizing high-dimensional structure with the incremental grid growing neural network. In: Proc. 12th int. conf. on machine learning, San Francisco, CA, USA, pp 55–63. citeseer.ist.psu.edu/blackmore95visualizing.html
8. Borner K, Chen C, Boyack K (2003) Visualizing knowledge domains. Annu Rev Inf Sci Technol (ARIST) 37:179–255
9. Chalmers M (1996) A linear iteration time layout algorithm for visualising high-dimensional data. In: VIS '96: Proc. of the 7th conf. on visualization. IEEE Comp. Society, Los Alamitos, pp 127–132
10. Chen C (2004) Information visualization: beyond the horizon. Springer, New York
11. Chen C (2006) Citespace II: detecting and visualizing emerging trends and transient patterns in scientific literature. J Am Soc Inf Sci Technol 57(3):359–377. doi:10.1002/asi.v57:3
12. de Oliveira JB (2009) Two algorithms for automatic page layout and possible applications. Multimed Tools Appl 43:275–301. doi:10.1007/s11042-009-0267-y
13. Hassan-Montero Y, Herrero-Solana V (2006) Improving tag-clouds as visual information retrieval interfaces. In: Proc. of the I int. conf. on multidisciplinary information sciences and technologies, InSciT2006. Mérida, Spain
14. Jolliffe IT (2002) Principal component analysis [electronic resource]. Springer, New York
15. Kaski S (1997) Data exploration using self-organizing maps. Ph.D. thesis, Finnish Academy of Technology
16. Kaski S, Honkela T, Lagus K, Kohonen T (1998) WEBSOM—self-organizing maps of document collections. Neurocomputing 21(1–3):101–117
17. Kruskal J (1964) Nonmetric multidimensional scaling: a numerical method. Psychometrika 29(2):115–129
18. Kruskal J, Wish M (1978) Multidimensional scaling. Sage, Newbury Park
19. Kruskal JB (1964) Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. Psychometrika 29:1–27. doi:10.1007/BF02289565
20. Law M, Jain A (2006) Incremental nonlinear dimensionality reduction by manifold learning. IEEE Trans Pattern Anal Mach Intell 28(3):377–391
21. Law M, Zhang N, Jain A (2004) Nonlinear manifold learning for data stream. In: Proc. SIAM data mining, pp 33–44
22. Lewis DD, Yang Y, Rose TG, Li F (2004) Rcv1: a new benchmark collection for text categorization research. J Mach Learn Res 5:361–397
23. Li Y, Shapiro L, Bilmes J (2005) A generative/discriminative learning algorithm for image classification. In: Proc. 10th IEEE int. conf. on computer vision ICCV'05, vol 2, pp 1605–1612
24. Lopes AA, Pinho R, Paulovich FV, Minghim R (2007) Visual text mining using association rules. Comput Graph 31(3):316–326. doi:10.1016/j.cag.2007.01.023
25. Malheiros V, Hohn E, Pinho R, Mendonca M, Maldonado JC (2007) A visual text mining approach for systematic reviews. In: ESEM '07: Proceedings of the first international symposium on empirical software engineering and measurement. IEEE Computer Society, Washington, DC, USA, pp 245–254. doi:10.1109/ESEM.2007.13
26. Morrison A, Chalmers M (2004) A pivot-based routine for improved parent-finding in hybrid MDS. Inf Visual 3(2):109–122. doi:10.1057/palgrave.ivs.9500069
27. Nurnberger A, Detyniecki M (2002) Visualizing changes in data collections using growing self-organizing maps. In: Proc. int. joint conf. on neural networks IJCNN'02, vol 2, pp 1912–1917
28. Paulovich FV, Minghim R (2006) Text map explorer: a tool to create and explore document maps. In: Proc. of the conf. on information visualization IV '06. IEEE Comp. Society, Washington, DC, pp 245–251. doi:10.1109/IV.2006.104
29. Paulovich FV, Nonato LG, Minghim R, Levkowitz H (2008) Least square projection: a fast high-precision multidimensional projection technique and its application to document mapping. IEEE Trans Vis Comput Graph 14(3):564–575
30. Paulovich FV, Oliveira MCF, Minghim R (2007) The projection explorer: a flexible tool for projection-based multidimensional visualization. In: Proc. XX Brazilian symp. on comp. graphics and image proc. SIBGRAPI 2007. IEEE Comp. Society, Washington, DC, pp 27–34. doi:10.1109/SIBGRAPI.2007.39
31. Pinho R, Oliveira MCF (2009) Hexboard: conveying pairwise similarity in an incremental visualization space. In: IV '09: Proceedings of the 2009 13th international conference information visualisation. IEEE Computer Society, Washington, DC, pp 32–37. doi:10.1109/IV.2009.12

32. Pinho R, Oliveira MCF, Minghim R, Andrade MG (2006) Voromap: a Voronoi-based tool for visual exploration of multi-dimensional data. In: IV '06: Proceedings of the conference on information visualization. IEEE Computer Society, Washington, DC, pp 39–44. doi:10.1109/IV.2006.131
33. Pinho R, Oliveira MCF, de Andrade Lopes A (2009) Incremental board: a grid-based space for visualizing dynamic data sets. In: SAC '09: Proceedings of the 2009 ACM symposium on applied computing. ACM, New York, pp 1757–1764. doi:10.1145/1529282.1529679
34. Rodden K, Basalaj W, Sinclair D, Wood K (1999) Evaluating a visualization of image similarity as a tool for image browsing. In: Proc. of the 1999 IEEE symp. on information visualization
35. Rodden K, Basalaj W, Sinclair D, Wood K (2001) Does organisation by similarity assist image browsing? In: Proc. SIGCHI conf. on human factors in computing systems, pp 190–197
36. Sammon JW (1969) A nonlinear mapping for data structure analysis. IEEE Trans Comput 18(5):401–409. doi:10.1109/T-C.1969.222678
37. Wise JA (1999) The ecological approach to text visualization. J Am Soc Inf Sci 50(13):1224–1233. doi:10.1002/(SICI)1097-4571(1999)50:13<1224::AID-ASI8>3.0.CO;2-4
38. Wong P, Foote H, Adams D, Cowley W, Thomas J (2003) Dynamic visualization of transient data streams. In: Proceedings IEEE information visualization, pp 97–104

Roberto Dantas de Pinho earned the PhD degree in Computer Science and Mathematical Computing in 2009 from the University of São Paulo, Brazil. He has a MSc. degree in energy industry regulation (2002) and a BSc. in Computer Science (1995), both from Universidade Salvador, Brazil. He has been a visiting research scholar at Drexel University in 2008/2009. He has significant experience in teaching, consulting and software development. He is currently a Science & Technology Analyst at the Brazilian Ministry of Science & Technology. He does research in Information Visualization, Visual Text Mining and Science & Technology policy. He is a member of the ACM and of the Brazilian Computer Society.

Maria Cristina Ferreira de Oliveira received the BSc in computer science from the University of São Paulo, Brazil, in 1985, and the PhD degree in electronic engineering from the University of Wales, Bangor, in 1990. She is currently a Professor at the Computer Science Department of the Instituto de Ciências Matemáticas e de Computação, at the University of São Paulo, Brazil, and has been a visiting scholar at the Institute for Visualization and Perception Research at University of Massachusetts, Lowell, in 2000/2001. Her research interests are in Visual Analytics, Visual Data Mining, Information Visualization and Scientific Visualization. She is a member of the ACM and of the Brazilian Computer Society and is currently the chief editor of the Journal of the Brazilian Computer Society.

Alneu de Andrade Lopes holds a PhD in Computer Science from University of Porto—Portugal (2001). Currently, he is an Assistant Professor at the University of São Paulo—Brazil. His main research interests lie on Artificial Intelligence, Machine Learning and Propositional and Relational Data Mining.