Info Vis

Article

Information Visualization
0(0) 1–17
© The Author(s) 2012
Reprints and permission:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/1473871612439644
ivi.sagepub.com
SAGE

Recommendation and visualization of similar movies using minimum spanning dendrograms

Michail Vlachos[1] and Daniel Svonava[2]

Abstract
Exploration of graph structures is an important topic in data mining and data visualization. This work presents a novel technique for visualizing neighbourhood and cluster relationships in graphs; we also show how this methodology can be used within the setting of a recommendation system. Our technique works by projecting the original object distances onto two dimensions while carefully retaining the 'backbone' of important distances. Cluster information is also overlayed on the same projected space. A significant advantage of our approach is that it can accommodate both metric and non-metric distance functions. Our methodology is applied to a visual recommender system for movies to allow easy exploration of the actor–movie bipartite graph. The work offers intuitive movie recommendations based on a selected pivot movie and allows the interactive discovery of related movies based on both textual and semantic features.

Keywords
Data embedding, data projection, minimum spanning tree, hierarchical clustering, distance preservation, simulated annealing

Introduction

Data visualization is an inherent task in data analysis and data mining, because it 'forces us to see' (in the words of John Tukey[1]) hidden or obvious properties and relationships of the data examined. The abundance of highly dimensional and complex data intensifies the need for advanced visualization techniques that are able to translate the original object relationships into a form easily comprehensible by the human brain, i.e. translated into two or three dimensions.

Despite the great proliferation of visualization techniques and tools, the majority of low-dimensional visualization techniques typically focus on preserving, in an approximate manner, all the object distances. In that sense they end up preserving exactly none of the original distances. The methodology presented in this work retains a subset of important distances very accurately: those belonging to the minimum spanning tree graph. This helps preserve the *local* data structure and partially also the *global* structure.

In addition, the method provides a novel way of capturing both object relationships and cluster information. Our approach fuses two-dimensional (2D) mapping techniques with dendrograms for resolving the cluster structure at different granularities. The dendrogram is overlayed (either literally or metaphorically) on the third dimension (i.e. extruding from the 2D page plane) to determine the cluster information. The user can visualize the cluster information at various granularities by projecting the cluster information onto the 2D mapping. The user can interactively select the cluster resolution, starting from small, low-level clusters and gradually advancing to higher-level cluster

[1]IBM Zurich Research Lab, Switzerland
[2]Slovak University of Technology, Bratislava, Slovakia

Corresponding author:
Michail Vlachos, IBM Research,

structures. Finally, the proposed mapping carefully considers both metric and non-metric distance functions.

Besides its usefulness for general data visualization, the proposed technique can offer meaningful ways of data exploration in the following **application** areas:

– **Business intelligence**. An important task of marketing teams of every large company is market segmentation, that is, identifying groups of customers with similar buying characteristics. Our technique can simplify tasks, such as revealing meaningful customer clusters, with the help of the interactive multi-resolution feature it offers.
– Expanding on the above notion, the proposed data viewing paradigm can also be fruitfully realized in the context of **recommender systems**. Consider, for example, the case of major corporations that offer a range of products and services to their customers. It is of great interest not only to identify groups of customers interested in certain products but also the 'order' of approach to those customers (e.g., from most likely to least likely to buy). By providing the neighbourhood graph, tasks like the aforementioned can be addressed in a more informative way.

This work focuses on using the proposed visualization technique for building a **movie recommendation engine** on a movie–actor bipartite graph, allowing the exploratory visualization of the graph space.

The structure of the paper is as follows. First, we review the related work and place our work in the proper context next to graph visualization tools and low-dimensional embedding techniques. We continue in the 'Description of our approach' section with an overview of the proposed mapping technique. We highlight what are the differences when metric or non-metric distance functions are used. We also explain how a placement technique inspired by simulated annealing preserves the mapping quality but helps reduce visual clutter. The 'Application' section illustrates an implementation of the proposed method within a movie recommender system; we provide a description of the prototype application that we have built. We give several visual examples of the recommendations and the mapping as provided by our system. The 'Experiments' section evaluates the distance preservation characteristics of our mapping. We also provide a comparison against ISOMAP.

Related work

Graph structures are present in most scientific disciplines; therefore, several graph visualization techniques have been presented over the years.[2–6] Excellent surveys of the various methods can be found in Refs. 7 and 8. Compared with previous work on graph visualization, the focus of this work is primarily not on scalability, but on quality of representation.

Our technique presents a mapping (or projection) f that preserves a subset of distances given a distance matrix D, which records pairwise distances of high-dimensional objects under a given distance function δ. The objects are originally described in a feature space R^n, but we assume we have access only to their pairwise distances. The distance function δ can be either metric or non-metric. Multi-dimensional scaling (MDS)[9] is one of the most well-known low-dimensional mapping techniques. Objects are placed on a lower-dimensional space so as on average the original pairwise distances are preserved as closely as possible.

In recent years, many visualization techniques focus on the topic of non-linear dimensionality reduction. One of the most popular methods is ISOMAP,[10] which works in a similar manner with MDS but focuses on preserving geodesic distances instead of the original distances. Therefore, the minimum spanning tree (MST) of the original distance matrix is computed and then the geodesic distances are estimated by computing the distance of each pair of objects *through* the MST-participating edges. These new distances are then projected onto a low-dimensional space using a Principal Component Analysis (PCA)-like approach. The above projection leads to an 'unfolding' of the high-dimensional manifold, hence promoting the visualization of complex structures on two or three dimensions. Other similar (in spirit) projection techniques include the Laplacian Eigenmaps,[11] Hessian Eigenmaps,[12] the Locally Linear Embedding[13] and others.[14,15] An overview of manifold projection techniques has been presented by Carter.[16] The goal of those projection methods is to preserve all distances approximately, while our approach preserves a subset of distances (spanning-tree distances) exactly. In the experiments we show that our methodology leads to reduced distortion of the original distances on the lower dimensionality.

Data-embedding techniques can also be used for visualizing high-dimensional data. Such methods embed arbitrary spaces into a Euclidean or pseudo-Euclidean space. Examples include the Bourgain embedding,[17,18] FastMap[19] and, recently, BoostMap.[20]

Locality-sensitive hashing (LSH) methods can also be applied to achieve a low-dimensional data projection.[21,22] LSH methods are generally applicable for distance functions where locality-sensitive functions have been identified, such as for real vector spaces with L_p distance measures, or bit vectors when compared with the Hamming distance. Therefore, LSH

techniques are not suitable for arbitrary distance functions. The above shortcoming can be lifted by employing *distance-based hashing* techniques.[23] In general, hash-based techniques are tailored for approximate nearest-neighbour retrieval rather than low-dimensional visualization. Moreover, only probabilistic guarantees are given with respect to the distance preservation. In comparison, our projection technique provides *deterministic* guarantees for metric distance functions, and *best-effort* distance preservation for non-metric approaches.

The work presented here constitutes an expanded version of the visualization technique presented in ref. 24. Here we explore in more detail how the proposed high-dimensional data embedding methodology can be used as the interface of a **movie recommendation engine**. Graph-based approaches are quite prevalent in the area of recommender systems.[25,26] Several recommendation systems have appeared in the literature in recent years, particularly for recommending videos[27,28] or movies.[29–32] Visual-driven interfaces have also been presented in the field of recommender systems.[33] Recently, researchers have been exploring the use of data-embedding approaches, particularly in the setting of collaborative filtering.[34] In our scenario, we use a movie–actor database as the underlying graph structure. Given a selected pivot movie the system can retrieve a set of similar movies that are portrayed and clustered on two dimensions. Our approach allows the exploratory visualization of the movie graph space and incorporates additional multimedia capabilities, such as automatic streaming of the selected movie trailers. In addition, the presented prototype showcases advanced filtering and selection functionalities. The final outcome is an interactive movie recommendation engine that allows the facile and meaningful exploration of the movie–actor bipartite graph.

Description of our approach

The proposed method combines the visual simplicity and comprehensibility of the MST with the grouping properties of hierarchical clustering approaches (dendrograms). Given pairwise distances describing the relationship of high-dimensional objects, the objective is to preserve a subset of important distances as well as possible on two dimensions, while at the same time accurately conveying the cluster information.

Our approach works as follows. First, we construct a minimum spanning tree layout on the 2D plane in such a way that all distances to a user-selected pivot point and the neighbourhood distances on the tree are exactly preserved. This construction carefully considers how to best portray the original object relationships for either metric or non-metric distance functions. Second, the dendrogram cluster hierarchy is constructed so that it can be positioned exactly *on top* of the MST mapping of objects. The cluster hierarchy can be frozen at any resolution level (tomographic view) to convey the multi-granular clusters that are formed. This concept is elucidated in Figure 1: objects are properly mapped on the 2D plane whereas on the third dimension is portrayed the hierarchy of the clustering structure. Naturally, this is only a conceptual

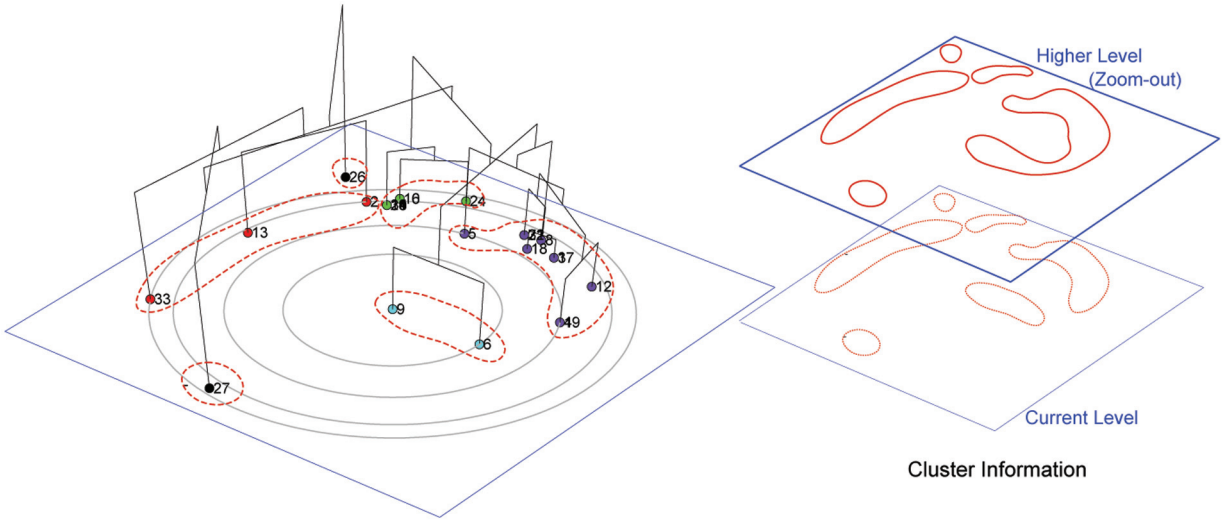

Figure 1. Conceptual illustration of our approach. The spanning tree structure is preserved and projected onto a 2D plane, while cluster structure is overlayed in the third dimension. 'Cutting' the dendrogram at a certain level projects the cluster structure onto 2D.

illustration of our approach. In practice, cluster information is also projected onto two dimensions, for example by properly colouring the nodes belonging to the same cluster. Therefore, by 'cutting' the derived dendrogram on a user-defined level, clusters are formed on the two-dimensional plane. Clusters are expanded or contracted appropriately, as the user drills up or down on the cluster hierarchy.

The advantages of the proposed technique include:

1. Preservation of both local and global structure.
2. Concurrent visualization of neighbourhood and cluster structure.
3. Multi-granular data visualization. The user can select the desirable detail of visualization (zoom in/zoom out capability). This is beneficial in its own right, when deciding, for example, the validity of a cluster instance.[35]
4. Ease of interpretation. Our technique adapts on widely utilized visualization primitives, with which users have interacted before. From a usability perspective, this is beneficial for the user, as many psychological studies clearly suggest that 'exposure to similar conditions lays the groundwork for quicker and deeper comprehension'.[36]
5. Our technique is *agnostic* on the distance function. The input to our technique can be any distance matrix between objects and the distance function can be either metric or non-metric (i.e. obey or not the triangle inequality).

Neighbourhood preservation

We will first explain how to capture on two dimensions the relationship between a set of high-dimensional objects. Not all pairwise distances can be preserved on two dimensions. In d dimensions one can preserve exactly d distances with respect to a given object.[10] Here, we choose to maintain, as well as possible, the spanning tree distances, because these distances offer valuable information about both the local relationships and the general global structure.[37,38]

We begin by constructing the spanning tree on the original high-dimensional objects. One object is selected as the pivot and mapped in the centre of the 2D plane coordinate system. By traversing the spanning tree, objects are positioned on the 2D plane by triangulating the distances to two objects: the pivot object and the neighbouring point previously mapped on the spanning tree.

Example. Suppose the first two points (A and B) of the MST have already been mapped, as shown in Figure 2(a). Let's assume that the second distance preserved per object is the distance to a reference point, which in our case is the first point. The third point is mapped at the intersection of circles centred at the reference points. The circles are centred at A and B with radii of $d(A, C)$ – the distance between points A and C – and $d(B, C)$, respectively. Owing to the triangle inequality, the circles either intersect in two positions or are tangent. Any position on the intersection of the circles will retain the original distances towards the two reference points. The position of point C is shown in Figure 2(a). The fourth point is mapped at the intersection of the circles centered at A and C (Figure 2(b)) and the fifth point is mapped similarly (Figure 2(c)). The process continues until all the points of the MST are positioned on the 2D plane, and the final result is shown in Figure 2(d).

The mapping technique presented will retain *exactly* the distances between all points and the pivot sequence, as well as between the nodes that lie at the edges of the spanning tree. This creates a powerful visualization technique that not only allows preservation of the nearest neighbour distances (local structure), but, in addition, retains distances with respect to a single reference point, providing the option of a global data view using that object as a pivot.

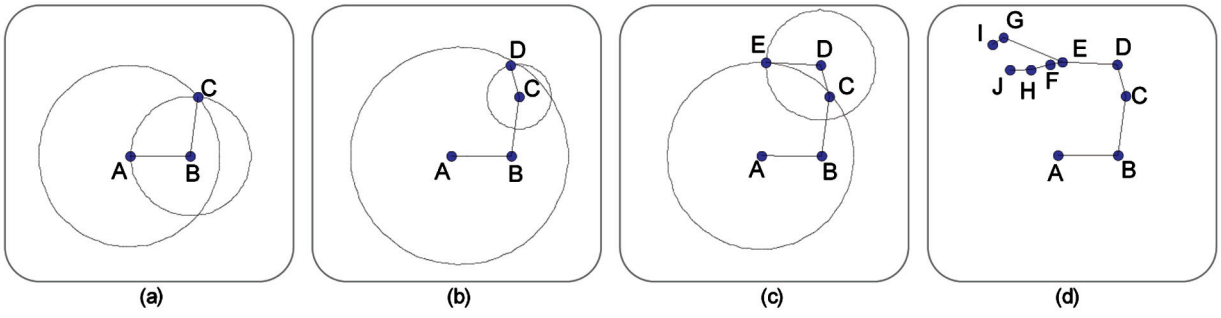

Figure 2. Two-dimensional embedding of the spanning tree of objects.

Algorithm 1 NodeMap(G, p) function

Require: $G = (V, E)$ is the original graph, (V, E_t) is the created MST, $p \in V$ is the pivot object, S is a stack
Ensure: $\forall v \in V$: Pos(v) $\in R^2$, $L_2(pos(v), pos(p)) = E_w(v, p)$
  Pos(p) $\leftarrow (0, 0)$
  //distribute children in a regular star-like shape
  for all $c_i \in children(E_t, p)$ **do**
    Pos(c_i) $\leftarrow (0, E(v, c))$
    $rotate(\text{Pos}(c_i), \frac{2\pi^* i}{|children(p)|})$
    $S.push(c_i)$
  end for
  //process the rest of the graph in DFS order
  while $|S| > 0$ **do**
    $t \leftarrow S.pop()$
    PositionChildren($p, t, children(E_t, t)$)
    for all $c_i \in children(E_t, t)$ **do**
      $S.push(c_i)$
    end for
  end while
  return Pos

Algorithm. We provide a pseudocode of the mapping procedure in Algorithm 1. The input to the mapping algorithm is the spanning tree on the given pairwise distances and the user-defined pivot object. The pivot object is positioned in the origin of the coordinate system. Subsequently, the algorithm positions its children in a star-like pattern, each in its corresponding distance to the pivot with uniform angular spacing between consecutive children. The remainder of the graph is processed in depth–first–search (DFS) order. Once a node is popped from the stack, all its children are positioned by triangulating the already mapped positions of the pivot and parent node. The `PositionChildrenfunction` (Algorithm 2) demonstrates the triangulation process. It calls the CircleCircleIntersection function, which returns the intersection points between two circles. Note that this version of the function refers to the case when a metric distance function is used and the circles are guaranteed to intersect. To handle the non-metric case, we introduce the proper extensions in subsequent sections.

The two circles will intersect in at most two points. The current pseudocode from the two candidate intersections selects the one with the lower azimuth value with respect to the currently expanded node. This is indicated by the `Left` position and defined in Algorithm 3. Once a node position is mapped on 2D, all its children nodes are pushed onto the stack, and the process repeats until no nodes remain in the stack.

Algorithm 2 PositionChildren(p, t, C)

Require: $G = (V, E)$, Pos(p), Pos(t) $\in R^2$, $C \subset V$
  // Pos(p) are the root (pivot) coordinates and Pos(t) are the parent coordinates
  // C is a set of children of the node t
Ensure: $\forall v \in C$: Pos(v) $\in R^2$
  // Pos() contains the 2D coordinates for all children nodes
  for all $c_i \in C$ **do**
    //find candidate points that preserve the distances E from the root and pivot nodes
    //by computing the intersection of the two distance circles
    $I \leftarrow$ CircleIntersect(Pos(p), $E(p, c_i)$, Pos(t), $E(t, c_i)$)
    //position the current child at the left candidate
    Pos(c_i) $\leftarrow$ Left(I, t)
  end for

Algorithm 3 Left/ Right(P, t)functions

Require: $\forall e \in P : e \in R^2$, Pos($t$) $\in R^2$
Ensure: point $e' \in R^2$ minimizes the azimuth of the vector $e' -$ Pos(t) for the Left variation of this function and maximizes it for the Right variation
  return $e' : min/max\{\text{Azimuth}(e' - \text{Pos}(t)) : \forall e' \in P\}$

Layout optimization using simulated annealing

The focus of the previously presented mapping approach was the preservation of distances on the spanning tree. However, when dealing with data visualization, other issues need to be addressed as well. One such issue is *clarity of display*,[39] in an effort to minimize the cognitive load.[40]

For achieving maximum visual clarity, it is advantageous to minimize the number of intersecting graph edges. Recall that when triangulating the position of a third point to its neighbour and the pivot, the algorithm proceeds by identifying the intersection between two circles. One can readily see this in Figure 3; unless the two circles are tangent, there are two positions in which a newly mapped point can be placed (we cover the non-metric case where circles do not later intersect).

The core node mapping algorithm as presented in Algorithm 1 uses the PositionChildren function, which greedily picks the intersection based on the azimuth. In order to select more intelligently which of the two mapping positions to follow, we adapt an optimization process that leads in reduction of the number of edge intersections. We employ a probabilistic global optimization technique based on *simulated annealing (SA)* principles.[41] SA is an effective optimization method when the search space is discrete, which is the case in our scenario of interest.

SA algorithms utilize 'heating' and 'cooling' primitives to avoid the situation of the objective function getting stuck at a local minimum. Heat causes random moves to states of higher energies, whereas a slower cooling provides chances for convergence to configurations with lower energy.

In each step the current solution found by the SA algorithm is randomly transformed to a nearby

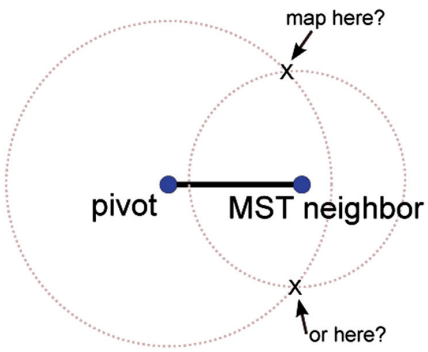

Figure 3. Selecting which of the two positions a new point is mapped to can affect the number of subsequent intersections with other edges . As the triangulation indicates two candidate positions for each point that needs to be mapped, there exist a large number of potential layouts that satisfy the triangulation mapping process.

solution in the state space. The magnitude of change is random but proportional to the temperature t. The newly generated candidate solution is scored by the objective function. If it performs better than the current solution, it is kept as the reference solution for the next iteration. Owing to the vast size of the state space in our setting, we do not permit the state expansion in a direction that does not improve the score.

In our setting, we need to decide which of the two positions of the intersecting circles to choose. Therefore, the decision state space is a binary vector of length N indicating which of the two intersection points (denoted as `Left` or `Right`) to choose for all N edges of the spanning tree. The current state in the decision space is stored as a vector CS. The function `PositionChildren` that calculates the node placement dictated by a given binary vector CS is outlined in Algorithm 4.

Algorithm 4 PositionChildren $\_SA(P, t, C, \mathrm{CS})$

Require: $G = (V, E)$, $\mathrm{Pos}(p), \mathrm{Pos}(t) \in R^2$, $C \subset V$, $\forall e \in E : \mathrm{CS}(e) = \mathrm{TRUE}|\mathrm{FALSE}$
// CS is a binary vector defining the mapping for a given tree
//keeping one bit of information for every Left/Right placement decision
Ensure: $\forall v \in C : \mathrm{Pos}(v) \in \mathbb{R}^2$
// Pos()contains the 2D coordinates for all children nodes
for all $c_i \in C$ **do**
$I \leftarrow \mathrm{CircleIntersect}(\mathrm{Pos}(p), E(p, c_i), \mathrm{Pos}(t), E(t, c_i))$
if $\mathrm{CS}(E(t, c_i)) = \mathrm{TRUE}$ **then**//query the control bit for a given edge
$\mathrm{Pos}(c_i) \leftarrow \mathrm{Left}(I, t)$
else
$\mathrm{Pos}(c_i) \leftarrow \mathrm{Right}(I, t)$
end if
end for

The function `CircleIntersect` produces the candidate intersection points for every mapped child node. The node is placed at the `Left` (least azimuth) intersection point if the value of the bit of CS that represents the decision for the current edge equals TRUE. Otherwise the node is positioned at the `Right` intersection (larger azimuth position).

Algorithm 5 SALayout function

Require: $G = (V, E)$, $p \in V$
Ensure: $\forall v \in V : \mathrm{Pos}(v) \in R^2$// Pos($v$)contains the 2D coordinates for all nodes in V
$\forall e \in E : \mathrm{CS}(e) \leftarrow \mathrm{TRUE}$
$T \leftarrow 1$//initial temperature
$\varepsilon \leftarrow$ desired numerical accuracy
while $T > \varepsilon$ **do**
$\mathrm{CS}_{\mathrm{new}} \leftarrow \mathrm{CS}$
for all $e \in E$ **do**
if $\mathrm{Random}[0, 1] < T$ **then**
$\mathrm{CS}_{\mathrm{new}}(e) \leftarrow \neg\, \mathrm{CS}(e)$
end if
end for
$\mathrm{Pos} = \mathrm{NodeMap\_SA}(\mathrm{CS})$
$\mathrm{Pos}_{\mathrm{new}} = \mathrm{NodeMap\_SA}(\mathrm{CS}_{\mathrm{new}})$
if $\mathrm{EdgeCrossings}(\mathrm{Pos}_{\mathrm{new}}) < \mathrm{EdgeCrossings}(\mathrm{Pos})$ **then**
//layout defined by $\mathrm{CS}_{\mathrm{new}}$ contains less edge crossings, move the search to $\mathrm{CS}_{\mathrm{new}}$
$\mathrm{CS} \leftarrow \mathrm{CS}_{\mathrm{new}}$
$T \leftarrow T^*\alpha$
else
$T \leftarrow T^*\alpha$
end if
end while
$\mathrm{Pos} = \mathrm{NodeMap\_SA}(\mathrm{CS})$

SA is performed on the control vector CS, directing the bit state position towards values that minimize the number of intersected edges of the spanning tree. A pseudocode for this process is given in Algorithm 5. During each iteration, the modified state vector $\mathrm{CS}_{\mathrm{new}}$ is evaluated by the layout algorithm and directed towards the state that minimizes the number of edge intersection. First, the node layouts Pos and $\mathrm{Pos}_{\mathrm{new}}$, dictated by CS and $\mathrm{CS}_{\mathrm{new}}$ respectively, are calculated using the function `NodeMap_SA`. To eliminate repetition we do not rewrite this function. It is identical with the one described in Algorithm 1, with the exception that

`PositionChildren` is replaced with PositionChildren_SA, which is its bit-vector driven counterpart as provided previously in Algorithm 4. Finally, function `EdgeCrossings` counts the number of intersecting edge pairs for a particular layout. If a candidate layout results in a lower number of intersected edges, it is kept for the next iteration. The temperature is increased ($T \leftarrow T^{*}\alpha$) so that additional opportunities can be explored and local minima potentially avoided. Conversely, if the candidate solution does not lead to a better solution, the global temperature is reduced ($T \leftarrow T/\beta$), with the cooling-off indicating that we may be close to reaching the global minimum.

In the experimental section, we show that the simulated annealing process significantly reduces the number of intersected edges on the 2D projected space.

Triangulation on non-metric distances

So far, it was assumed that the underlying distance measure obeys the triangle inequality; therefore, the circles around the reference points are guaranteed to intersect. However, many widely used distance functions (e.g. dynamic time warping,[42] longest common subsequence[43]) violate the triangle inequality, and thus the corresponding reference circles may not necessarily intersect. In such cases, one needs to identify the position of where an object should be placed with respect to the two circles, in such a way that the object is mapped as close as possible to the circumference of both circles. We need to identify the locus of points that minimize the sum of distances to the perimeters of two circles. One can show that the desired locus always lies on the line connecting the centres of the two circles.

We highlight necessary extensions that allow its proper usage under non-metric distances. We can identify two cases for the non-intersecting circles:

– Case 1: the two circles are disjoint and one does not enclose the other; and
– Case 2: one circle encloses the other.

One can show the following:

Lemma 1 *The point that minimizes the sum of distances to the perimeters of two circles lies on the line L connecting the two circle centres and is located midway on the line segment with vertices at the intersection of L with the circles' perimeters.*

Case 1 is shown in Figure 4. The midpoint C is given by the equation:

$$C = A_1 + \frac{A_2 - A_1}{d} \cdot \frac{d + r_1 + c \cdot r_2}{2} \qquad (1)$$

where $c = -1$, A_1 is the circle with larger radius, $d = |A_2 - A_1|$ and r_1, r_2 are the radii of the corresponding circles.

For case 2, where one circle encloses the other, one can identify two subcases:

– When the two circles have disjoint centres, as shown in Figure 5(a), then the coordinates of the sought point C that minimizes the distance to the two circles is also given by Equation (1), where in this case $c = 1$.
– When the two circles have common centres, then there exist two points per line that satisfy the distance minimization property, as shown in Figure 5(b). There is an infinite number of such lines, all passing through the common centre of the two circles.

$$C = A_1 + \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{bmatrix} \cdot (0, \frac{r_1 + r_2}{2}),\ \alpha \in [0, 2\pi]$$

The above modifications allow the positioning of the objects on the 2D plane, so that our mapping approach also accommodates cases where a non-metric distance function has been used to construct the input distance matrix.

The derived formulae are combined to implement the function `CircleCircleIntersection_NM` in Algorithm 6. This function can replace inside `PositionChildren` the metric case of `CircleCircleIntersection` algorithm, when the utilized distance measure is non-metric.

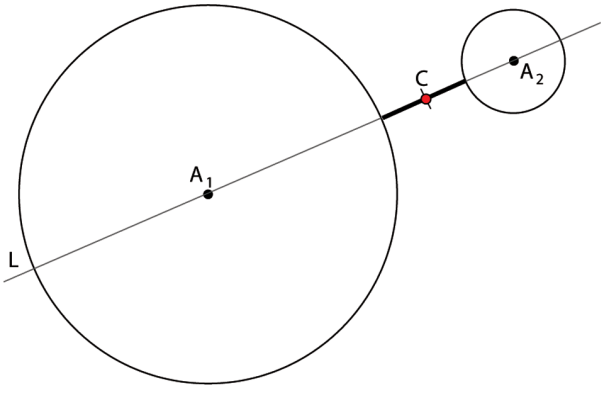

Figure 4. Circles that are disjoint.

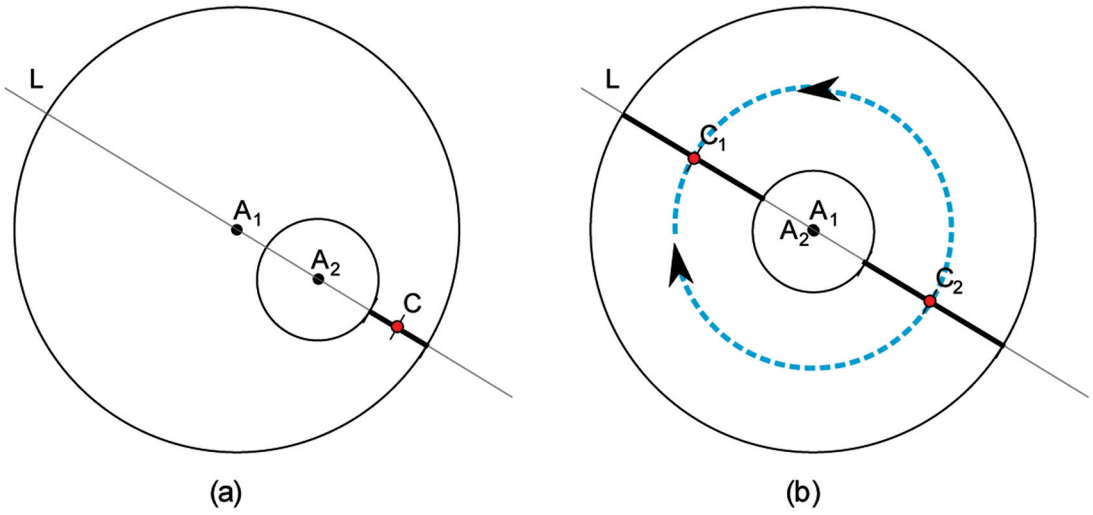

Figure 5. One circle encloses the other.

Algorithm 6 CircleIntersect_$NM(c_1.c, c_1.r, c_2.c, c_2.r)$ function

Require: $c_1.c \in \mathbb{R}^2$, $c_1.r \in \mathbb{R}$, $c_1.c \in \mathbb{R}^2$, $c_2.r \in \mathbb{R}$
Ensure: $(x, y) \in \mathbb{R}^2$
  $\epsilon \leftarrow$ desired numerical accuracy
  $D \leftarrow L_2(c_1.c, c_2.c)$
  if $D < \varepsilon$ **then**
    //circles are concentric
    return $(0, \frac{c_1.r + c_2.r}{2})$
  else if $D > c_1.r + c_2.r$ **then**
    //circles c_1 and c_2 do not intersect, nor overlap
    return $c_1.c + c_2.c - c_1.c/D^* c_1.r - c_2.r + D/2$
  else if $D < |c_1.r - c_2.r|$ **then**
    //circles c_1 and c_2 do not intersect, but one is contained in the other
    return $c_1.c + c_2.c - c_1.c/D^* c_1.r + c_2.r + D/2$
  else
    //triangle inequality holds, use regular circle intersection
    return CircleIntersect$(c_1.c, c_1.r, c_2.c, c_2.r)$
  end if

Cluster preservation

We now turn our attention on how to embed cluster information on the already mapped MST graph. Recall that the input for the algorithm is a matrix of pairwise distances. Based on the given pairwise distances one can build a hierarchical dendrogram. The dendrogram construction is based on a single linkage approach that merges *closest* singleton objects and clusters. The process iterates as follows:

1. The two closest items objects are merged together into one cluster and these merged objects are deleted.
2. The distances of the remaining objects to the newly formed cluster are recomputed. The distance is selected to be the *minimum* distance of an object to any of the cluster objects. This step is important because it allows us to exactly overlay the resulting dendrogram.
3. The process is repeated until only one cluster object remains.

The hierarchical cluster construction that we adapt updates the distances in such a way so that minimum distances between objects are favoured in the merging process. This allows us to effectively combine the two techniques. One can show the following lemma:

Lemma 2 *The merging order of the minimum spanning tree algorithm is the same as the merging order of the single linkage hierarchical clustering approach.*

What the above essentially tells us is that we can achieve a **perfect overlay** of the constructed hierarchy on top of the MST-mapped points, as the clustering order is the same as the order crystallized in the spanning-tree mapping. This fact can be also seen in Figure 1 where the single linkage dendrogram is positioned exactly on top of the spanning-tree mapping.

The above observation, combined with the hierarchical cluster information, provides the user with the ability to have multi-granular cluster view on two dimensions, by interactively imposing variable thresholds on the resulting dendrogram. This allows the creation of a 'tomographic view' of the clusters, and promotes the interactive formation of incrementally larger or smaller cluster structures. The concept is illustrated in Figure 6. By cutting the dendrogram using a smaller threshold, six clusters are created on the left. Imposing

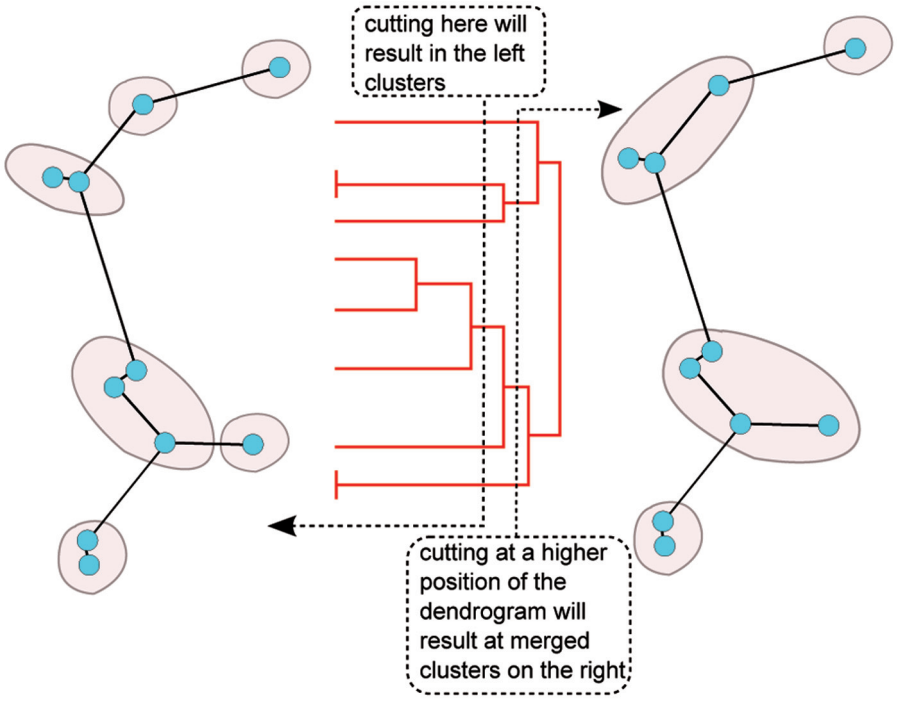

Figure 6. Cluster information conveyed by variable thresholds on a dendrogram information, and thus imposing 'zoom in' and 'zoom out' in the cluster structure.

progressively higher thresholds leads to merging of the clusters, as shown on the right side of the same figure.

Clusters can be conveyed on two dimensions either enclosed in ellipses or by forming the convex hull of the contained objects; another approach is through the means of colour. In our prototype implementation we convey cluster information by properly colouring the node perimeter and also the connected edges.

Overall complexity. Naively, the proposed algorithm comprises two stages: the hierarchical clustering and the creation of the spanning tree. When the single-linkage clustering (SLC) and MST algorithms are used, these stages can be performed separately and as the algorithms are inherently similar and thus produce compatible output, the resulting dendrogram can be positioned over the 2D-mapped spanning tree.

Upon closer inspection of the clustering algorithm, one can realize that the edges that initiate the cluster merging in the SLC algorithm are equivalent to the spanning tree produced by MST. This implies that there is no need to perform the actual MST algorithm on the original graph. The runtime complexity of the layout algorithm presented in Algorithm 1, dominated by the depth-first traversal of the spanning tree, is $O(n)$. The resulting complexity of our approach is therefore dominated by the clustering algorithm itself, which, in the case of SLC, is $O(n^2 \log n)$ where the $O(\log n)$ component is due to the operation of the Disjoint-Set data structure. The complexity of the latter component can be reduced to $O(\alpha(n))$, where α is a certain inverse of Ackermann's function, as shown by Chazelle.[44] Many graphs with high node cardinality, where the $O(n^2)$ component could impose a problem, are in practice sparse (e.g. social networks) where there can be established tighter bounds on the number of edges as a constant multiple of graph node count, for example $O(e) = O(k^* n) = O(n)$. Then, the overall running complexity is $O(n\alpha(n))$. Finally, Karger et al.[45] presented a randomized MST algorithm that runs in linear time with high probability using only edge-weight comparisons.

The mapping and clustering algorithm can be combined with the simulated annealing layout optimization ('Layout optimization using simulated annealing' section) to improve the visual clarity. The process of clustering and spanning tree discovery is separate from the layout algorithm. Therefore, the resulting complexity is $O(O_{orig} + n^* I)$ where O_{orig} is the original complexity discussed above and I is the number of iterations of the layout algorithm required by the optimization method to converge. In the 'Experiments' section we demonstrate that for graphs with approximately 100 nodes it is reasonable to expect low cardinality of iterations in the order of 100 – 200 for the SA portion of the code.

In practical applications the proposed visualization technique is not applied on an arbitrary number of graph nodes, because this would pose a screen flooding problem. The minimum spanning dendrogram is typically constructed as the outcome of a k-nearest-neighbour (k-NN) search with respect to the examined pivot. For k-NN operations an index structure (B-tree or R-tree) is typically employed, allowing for the expedient retrieval of the k-neighbourhood with respect to the pivot point. Therefore, the final visualization is ultimately constructed on top of a few hundred objects. This allows the very effective runtime operation of our solution.

Application: visualizing a movie–actor graph

Using the proposed methodology we have built a movie recommendation engine that visualizes a graph of related movies based on the pivot movie selected.

We use a movie graph comprising approximately 125,000 movies and 950,000 actors. Attributes of the movies include features such as title, year, genre, director, plot, language and cast, to name only a few. Each movie is enriched with additional textual features extracted from the internet. Therefore, each movie record is represented by a set containing keywords/phrases pertaining to the movie.

To evaluate the similarity between two movies, we consider how similar are the textual sets describing these movies. We compute the distance d using the Jaccard coefficient $\mathcal{J}$ for the textual sets of movies x and y:

$$d = 1 - \mathcal{J}(x,y) = 1 - \frac{|x \cup y|}{|x \cap y|}$$

The above distance is a metric distance function.[46,47] Other functions for evaluating the similarity, such as cosine similarity[48] or Latent-Semantic-Analysis (LSA)[49] can also be directly accommodated. However, the above simple approach worked very efficiently in practice and provided intuitive results, while having the added benefits of efficient execution. In addition, the Jaccard metric can work well even when comparing objects with different numbers of attributes, such as in our scenario (variable set of textual features per movie).

Graphical user interface and functionalities. The data visualization is accommodated through a graphical web interface (Figure 7), which incorporates the neighbourhood and cluster visualization technique described in the preceding sections. The interface allows the user to easily search for a movie with the aid of an autocomplete interface. It subsequently searches for the k-NN neighbours of the selected movie, where the parameter k is user selectable. For the set of neighbours the proposed visualization graph is created, placing the movie selected as the centre (pivot) object. The user can then easily identify other relevant movies, with the option to retrieve detailed information about the movie, such as participating actors or the movie plot. The user can easily modify the number of displayed movie clusters or even watch the movie trailer which is automatically retrieved from the Internet. The application allows the exploration of both sides of the

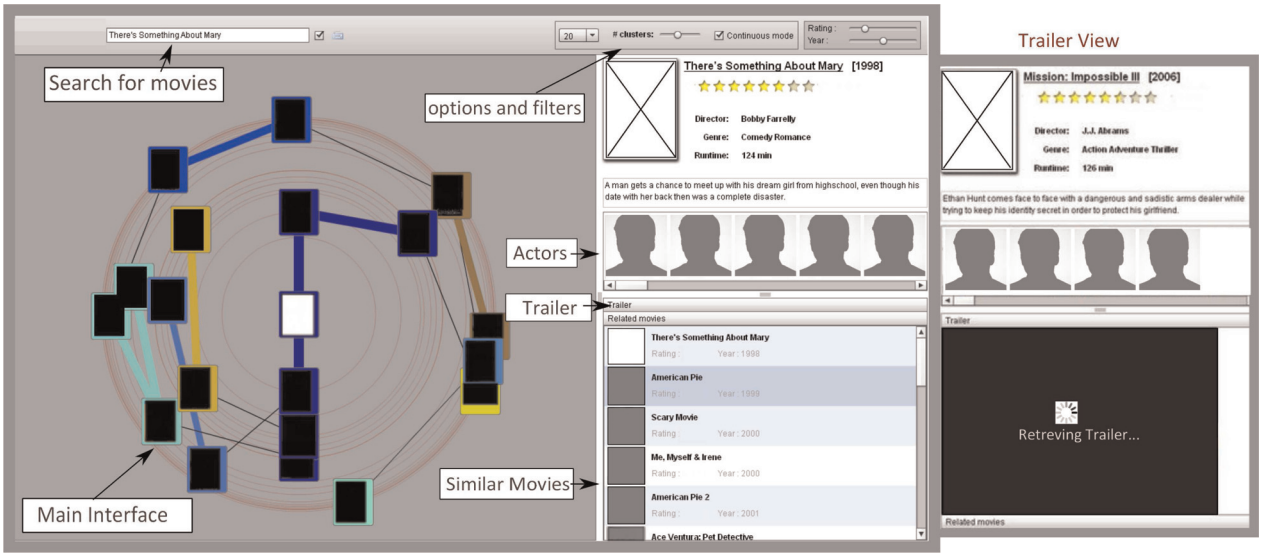

Figure 7. The web-based graphical user interface of the movie recommender system application.

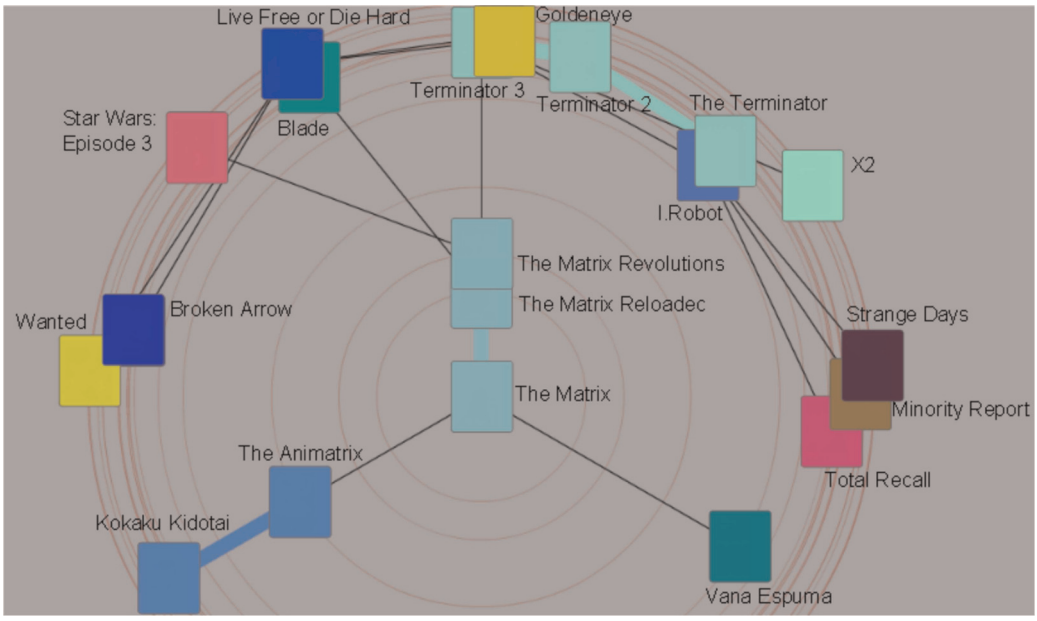

Figure 8. *The Matrix* selected as pivot object for our mapping.

movie–actor bipartite graph: either by deeper exploration of the movie graph (by clicking on a movie) or by searching/filtering for movies of a particular actor (by clicking on an actor's image).

Additional filtering functionalities include:

– filtering by rating, for example when interested in retrieving movies with a rating $> X$; and
– filtering by year, when the user is interested only in recent movie releases.

Below we provide specific examples of our mapping, highlighting its ability to be used as an effective and interactive movie recommendation system. Clusters are conveyed using varying border and edge colours. In practice, obviously movies could be illustrated by their DVD covers.

Examples. Selecting *The Matrix* as the pivot movie, our paradigm will correctly pack *The Matrix 2* and *The Matrix 3* to it (Figure 8). An adjacent cluster contains the 'Animatrix', an animé movie inspired by the *Matrix* trilogy. We can also identify a cluster of movies belonging to *The Terminator* series of movies. Other relevant movies that can be found on the graph include *Minority Report*, *Strange Days*, *Total Recall* and *Blade*, among others.

Placing *Titanic* as the pivot movie produces the 2D mapping shown at the top of Figure 9. The user can navigate through the graph and identify similar movies. Clustered with *Titanic* are the movies *Titanic* (1953), *Poseidon* and *Shakespeare in Love*. Another cluster displayed, shown in light green, includes movies such as *The Notebook*, *Atonement* and *Purple Rain*; all romantic drama movies.

Similarly, selecting *Star Wars* as the pivot movie correctly packs closely the remaining *Star Wars* movies (Figure 9, bottom). An adjacent cluster includes parodies of *Star Wars*, such as *Spaceballs* or *Thumb Wars*. Other related movies shown on the graph include adventure films such as *The Lord of the Rings* trilogy or science-fiction action thrillers such as *Aliens* and *Star Trek*.

Experiments

We evaluate how the SA layout algorithm reduces the number of edge intersections. We also highlight the ability of our algorithm to preserve the original object distances and provide a thorough comparison with the mapping provided by the popular ISOMAP algorithm.[10]

SA layout algorithm

We assess the quality of the node layout placement with and without the proposed SA layout algorithm. Recall that the SA algorithm uses an optimization function to properly place the mapped nodes, so as to minimize the number of intersected edges (Figure 10).

We use the dataset from the 'Application' section, which represents each movie with a bag of words. From this set we randomly select 1000 objects. For

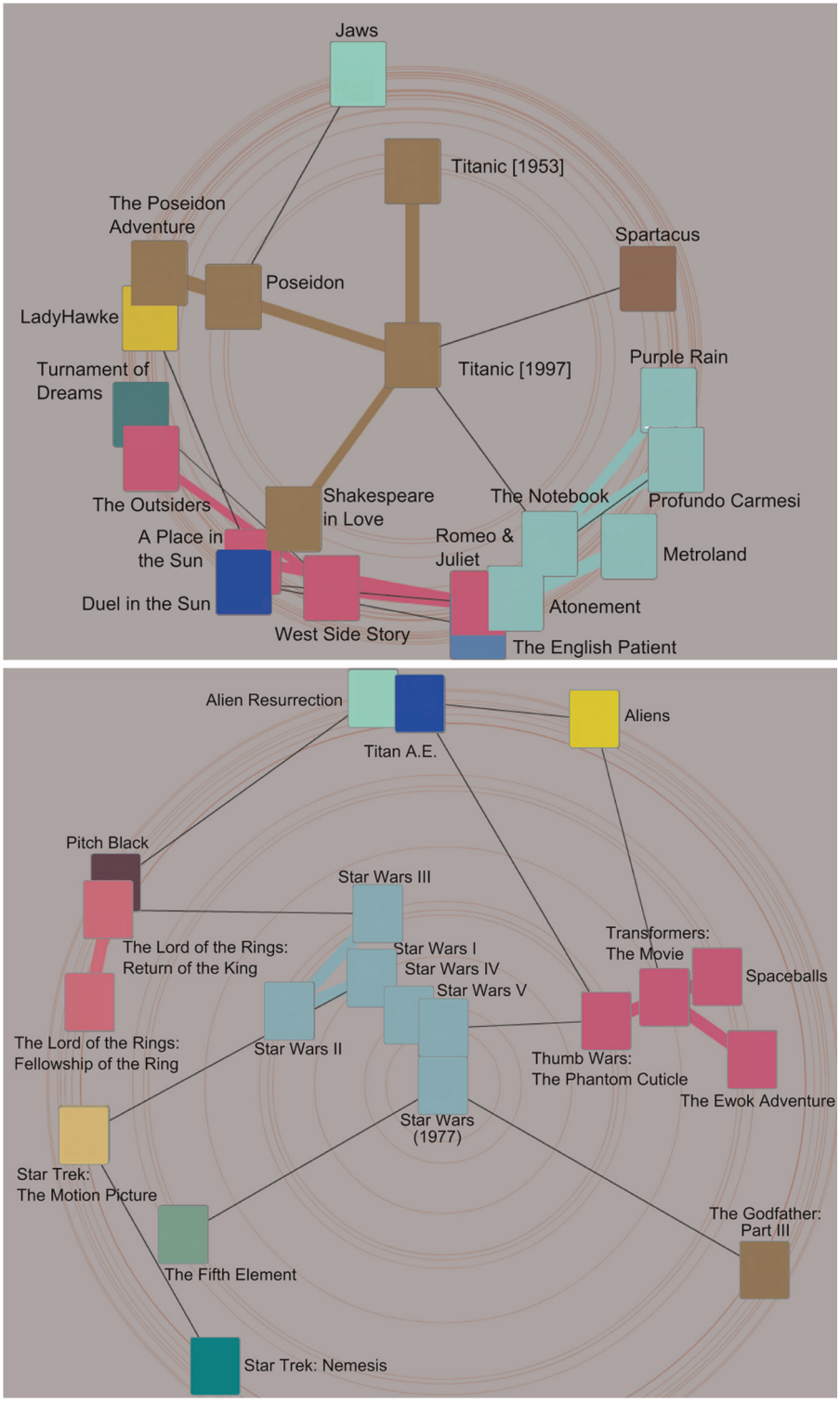

Figure 9. Using the proposed mapping on a movie graph. *Top:* The pivot movie is *Titanic*. *Bottom: Star Wars* is selected as the pivot movie.

each we retrieve the k-NNs for k equal to 20, 30, 40 and 50. The distance is computed using the Jaccard coefficient as described in the 'Application' section.

We consider each of the objects/movies as pivots and use the proposed method to visualize the relationship of the retrieved neighbourhoods, with and without the

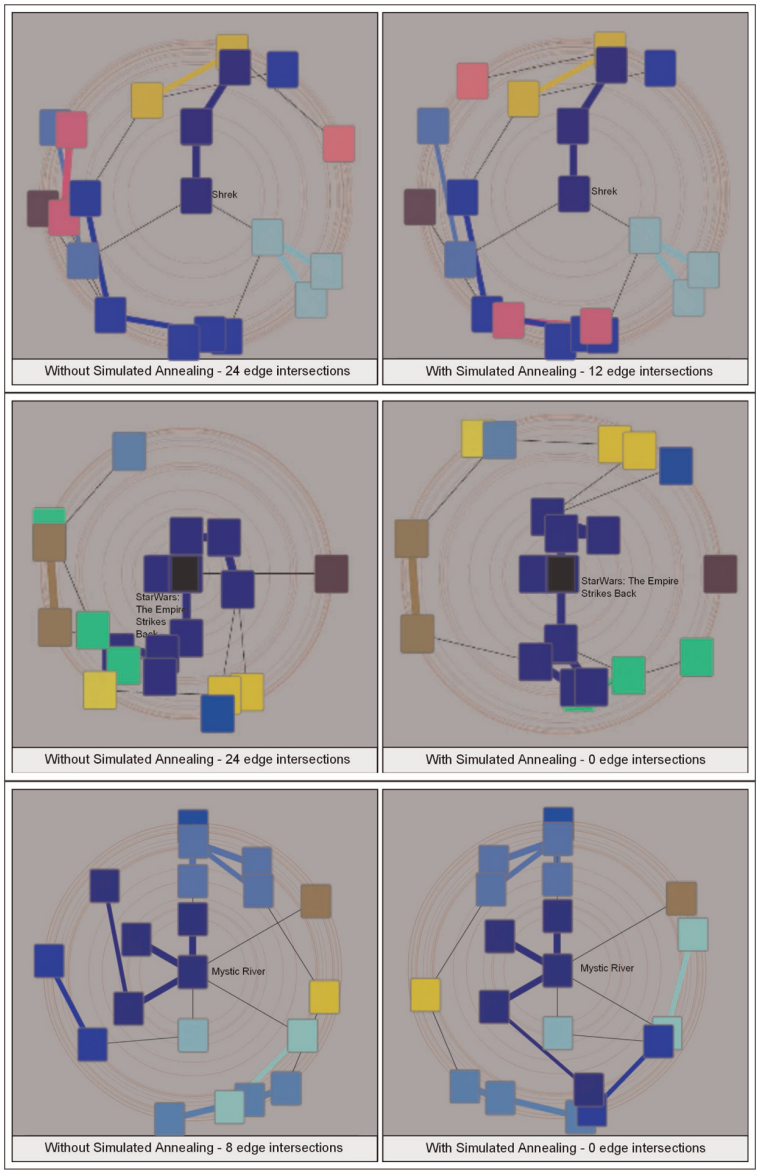

Figure 10. Examples of how the simulated annealing component improves the visual layout and reduces the number of edge intersections.

SA layout placement. The SA algorithm significantly reduces the number of intersected edges. This leads to better readability of the graph of mapped objects. Naturally, the SA algorithm incurs additional computational expense, because it introduces an additional number of iterations. The additional time required by the SA process is given in Table 1 (note that the SA component is executed within the web browser, and therefore the reader should emphasize the relative times for different k values rather than the absolute times). The table also reports the median number of edge intersections and the number of iterations for the layout algorithm, with and without the annealing component.

Table 1. Node placement using simulated annealing (SA) significantly reduces the number of edge intersections (median number shown). We also report the additional time incurred by the SA process.

k-value	Edge intersections		SA iterations	SA time
	Without SA	With SA	(median [min–max])	(sec)
20	36	12	102 [0–248]	0.4
30	108	48	146 [59–335]	1.4
40	222	112	146 [59–335]	4.3
50	346	212	146 [59–349]	4.4

We provide the median, minimum and maximum number of iterations. The median number of iterations ranges between 100 and 150 iterations, indicating a low complexity for the SA component. We note that both variations provide the same distance preservation with respect to the pivot and the MST distances. However, the SA implementation clearly enhances the visualization capacity of the mapping by providing a more intelligent node placement.

Distance preservation in projected space

The mapping presented in this work has been designed to accurately preserve on two dimensions the following two distance types:

– the distance of all points to the pivot (center) object; and
– the distance of all neighbouring points on the MST.

In addition to the above cases, we also evaluate the distortion of the distance for all pairs of objects. This is a criterion that our approach does not directly optimize. However, because we preserve several important distances exactly, we expect that the remaining ones do not deviate significantly.

We also put forward a comparison with the 2D mapping provided by ISOMAP. ISOMAP represents a method for performing a non-linear mapping of D-dimensional points, on a lower dimensionality d. It has attracted significant interest from the data-mining and machine-learning community because of its effectiveness and simplicity of implementation. The mapping performed by ISOMAP attempts to 'unfold' the data, when they lie on a high-dimensional manifold. It shares some commonalities with the approach presented in this work, because ISOMAP achieves the data unfolding by computing spanning-tree distances in the original space, and preserving them on the d-dimensional space through a PCA-like projection operation.

However, the process of preserving on average *all* spanning tree distances eventually is the weak point of ISOMAP. The final projection, even though revealing about the overall data structure, eventually results in significant distortions of the original distances. Here, we show that while our mapping preserves perfectly the pivot and neighbouring MST distances, ISOMAP results in a projection of lower quality.

Experiment. From our movie database we pick randomly 1000 movies as pivot objects and then retrieve the $k = 20,50,100$ nearest neighbours, as suggested by the distance function. For each object, we evaluate how close the projected distances are on two dimensions, compared with the original high-dimensional ones. We identify three subsets of distances: (a) distances of all objects to the pivot, (b) distances of neighbouring spanning-tree distances and (c) all pairwise distances.

We make the above distinction because our approach carefully preserves the first two types of distances. ISOMAP requires one parameter as the input: the neighbourhood for creating the local graph of each object. Selecting this parameter is not an easy task, because it is data dependent. Typically, low values that lead to a fully connected graph are preferable. For each experiment we set the neighbourhood value to $k/3$, because lower values led to disconnected graph components.

Our findings are reported in Figure 11. Distance preservation is calculated as the mean ratio of distances in the projected space over distances in the original space. That is, if M is the original matrix of pairwise distances and $\hat{M}$ are the distances on the 2D space, then we estimate the ratio $r = mean(M/\hat{M})$. Values of r closer to 1 indicate that the projected distance on two dimensions is closer to the original distance. The ratios r for the corresponding 1000 graphs are sorted from the lowest to the highest values.

The experiments indicate that distances to the pivot object are accurately preserved; the distance preservation reaches 100% (Figure 11, top left). Distances on the MST graph are also perfectly retained, as indicated on the top-right graph of Figure 11. This is not surprising, because our method performs the mapping in such a way, so that pivot and MST distances are preserved. In contrast, ISOMAP results in significantly lower performance regarding preservation of original MST distances.

Finally, we evaluate what is the accuracy of our technique for preserving all pairwise object distances. Our methodology does not directly minimize the distortion of all pairwise distances on 2D. However, by preserving the spanning tree distances, our methodology essentially preserves the '*backbone*' of the dataset structure. Therefore, the remaining distances do not generally deviate significantly from the original distances.

This result is conveyed on the bottom-left graph of Figure 11. As expected, both our mapping and ISOMAP do not perfectly preserve pairwise distances. Because from this graph it is not apparent which approach provides a better mapping, we present the same results on a win–lose graph (bottom-right side). For each of the 1000 experiments this graph indicates when a mapping is closer to the original distance. For $k = 20$, our approach provides a better mapping than ISOMAP for 45% of the cases. For larger neighbourhoods this situation reverses. Our methodology is better for 75% and 87% of the experiments for $k = 50$ and $k = 100$, respectively. This final result implies that for large graph instances ISOMAP mapping provides a lower quality of mapping, because as the algorithm

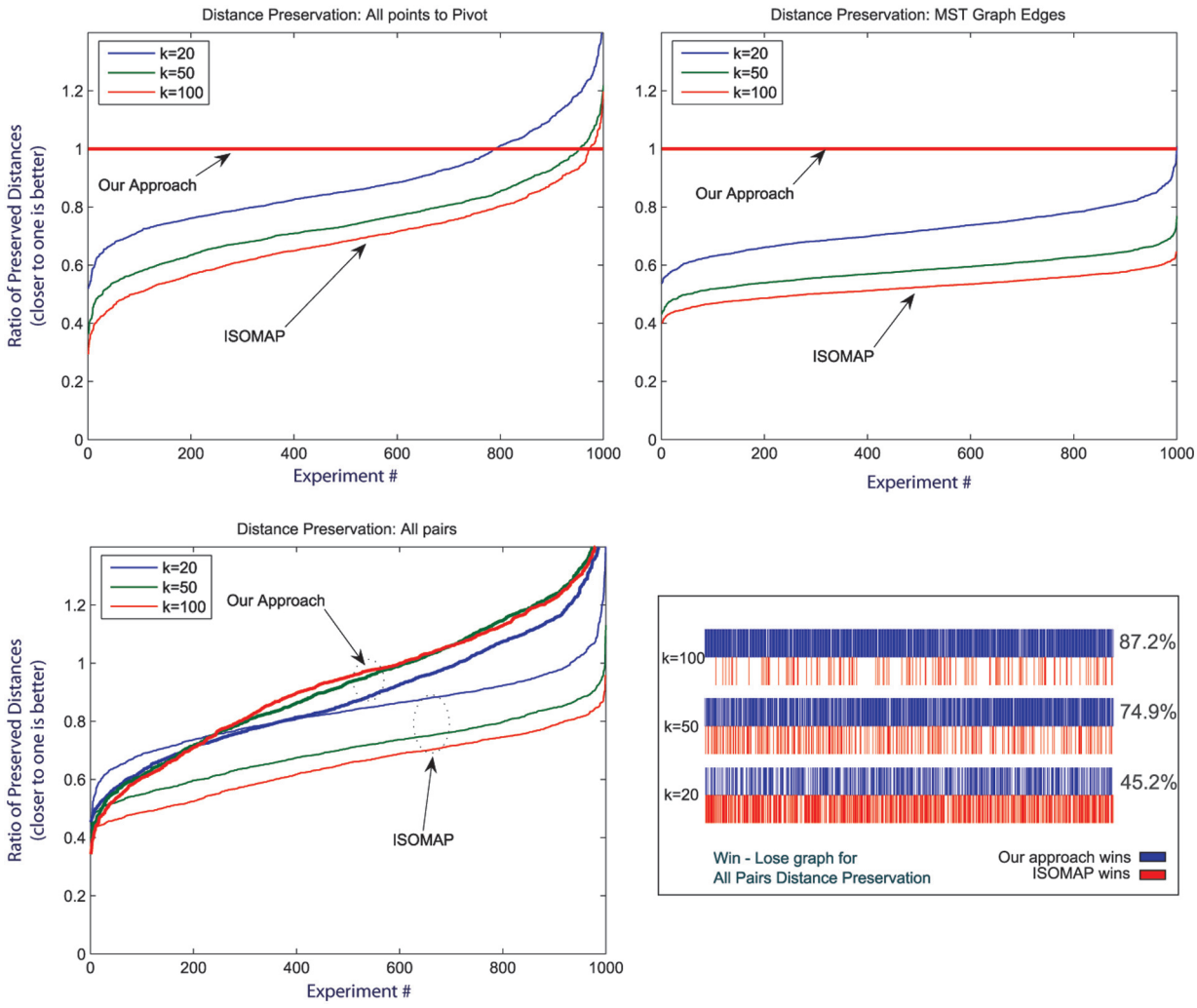

Figure 11. Comparison with ISOMAP with respect to distance preservation in the projected space. **Top left:** Our approach exactly preserves all distances to the pivot object. **Top right:** Spanning distances are also very accurately preserved. **Bottom left:** Pairwise distances are generally better preserved by ISOMAP. However, for small neighbourhoods (k = 20), our approach results in an embedding that is only 2% worse than ISOMAP. **Bottom right:** win–lose graph for all-pair distance preservation.

attempts to preserve all distances it eventually distorts the original graph structure.

In conclusion, these experiments attest to the ability of our mapping technique to accurately preserve on low dimensions the original high-dimensional graph structure. We have verified empirically the high quality of mapping, in particular compared with other popular graph projection techniques, such as ISOMAP.

Current limitations and future work

The proposed mapping algorithm does not preserve all pairwise distances, but only distances to the central (pivot) object, as well as the neighbouring distances on the MST. Therefore, points that are distant on the MST graph have been mapped with respect to different neighbouring points. No guarantees can be made for the distance of these points on the 2D mapping. See, for example, in Figure 8 the movies *Star Wars: Episode 3* and *Live Free or Die Hard*. A tentative solution for this could be the following: as mapped points get more distant from the pivot, the algorithm can position the objects by preserving the weighted distance towards multiple points. This approach, of course, leads again to approximate distance preservation. Another related shortcoming of the technique is that even though it promotes exact distance preservation, this may hinder visualization, for example at highly clustered areas. A way to mitigate this could be

to relax the constraint of exact distance preservation. Then, the projected points of the spanning tree would be adjusted in order to promote better filling of the space. By considering a histogram of the distances of all objects toward the pivot object, this could be addressed as a histogram equalization process.

Conclusion

This work presented a new visualization paradigm for graphs and high-dimensional data, in general. Our approach combines both neighbourhood preservation and cluster preservation. Neighbourhood distances are virtually identical to the original high-dimensional distances with respect to a designated pivot point. We put forward several node layout policies, including a probabilistic SA placement algorithm, which retains the same distance preservation properties, while minimizing the number of intersected edges in the graph structure. This significantly enhances the visualization power of the lower-dimensional projected mapping. Finally, we have shown that our approach can effectively support both *metric* and *non-metric* distance functions.

A unique characteristic of our approach is that it exploits the commonalities between spanning-trees and single linkage hierarchical dendrograms, thus being able to capture on the projected graph both neighbourhood and cluster structure. In this manner our technique allows the interactive growth and shrinking of clusters at progressively granular levels. This facilitates the effective zooming in and out of the identified data clusters.

We have shown that the proposed technique offers superior mapping capabilities in comparison with previously used visualization techniques, such as ISOMAP. We have applied our visualization approach on a movie–actor graph, and built an interactive movie recommendation engine that showcases the above principles.

Funding

The research presented here was partially supported by a Marie Curie Reintegration Grant.

Acknowledgement

This work was conducted while the second author was visiting the IBM Zurich Research Lab.

References

1. Hoaglin DC, Mosteller F and Tukey JW. *Understanding robust and exploratory data analysis*. Wiley, New York; 1983.
2. Wattenberg M. Visual exploration of multivariate graphs. In: *Proceedings of the SIGCHI conference on human factors in computing systems*, 2006, pp.811–819.
3. Trethewey P and Höllerer T. *Interactive manipulation of large graph layouts*. Technical Report, University of California, Santa Barbara, CA, USA, 2009.
4. Dwyer T. Scalable, versatile and simple constrained graph layout. *Comput Graph Forum* 2009; 28(3): 991–998.
5. Gretarsson B, O'Donovan J, Bostandjiev S, et al. Smallworlds: visualizing social recommendations. *Comput Graph Forum* 2010; 29(3): 833–842.
6. Wong PC, Foote H, Mackey P, et al. A dynamic multiscale magnifying tool for exploring large sparse graphs. *Inform Vis* 2008; 7(2): 105–117.
7. Herman I, Melancon G and Marshall M. Graph visualization and navigation in information visualization: a survey. *IEEE Tran Vis Comput Graph* 2000; 6(1): 24–43.
8. Cui W. *A survey on graph visualization*. PhD Qualifying Exam (PQE) Report, Computer Science Department, Hong Kong University of Science and Technology, 2007.
9. Kruskal J and Wish M. *Multidimensional scaling*. SAGE, 1978.
10. Tenenbaum JB, de Silva V and Langford JC. A global geometric framework for nonlinear dimensionality reduction. *Science* 2000; 290(5500): 2319–2323.
11. Belkin M and Niyogi P. Laplacian Eigenmaps for dimensionality reduction and data representation. *Neural Comput* 2003; 15(6): 1373–1396.
12. Donoho DL and Grimes C. Hessian Eigenmaps: new locally linear embedding techniques for high-dimensional data. *Proc Nat Acad Sci USA* 2003; 100(10): 5591–5596.
13. Roweis S and Saul L. Nonlinear dimensionality reduction by locally linear embedding. *Science* 2000; 290(5500): 2323–2326.
14. Yang L. Distance-preserving projection of high-dimensional data for nonlinear dimensionality reduction. *IEEE Trans Pattern Anal Mach Intell* 2004; 26(9): 1243–1246.
15. Lee J and Verleysen M. Nonlinear dimensionality reduction of data manifolds with essential loops. *Neurocomputing* 2005; 67: 29–53.
16. Carter KM. *Dimensionality reduction on statistical manifolds*. PhD Thesis, University of Michigan, USA, 2009.
17. Bourgain J. On Lipschitz embeddings of finite metric spaces in Hilbert space. *Israel J Math* 1985; 52: 46–52.
18. Hristescu G and Farach-Colton M. *Cluster-preserving embedding of proteins*. Technical Report 99-50, Computer Sscience Department, Rutgers University, New Brunswick, NJ, USA, 1999.
19. Faloutsos C and Lin KI. FastMap: a fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In: *International conference on management of data (SIGMOD)*, 1995, pp.163–174.
20. Athitsos V, Alon J, Sclaroff S, et al. BoostMap: an embedding method for efficient nearest neighbor retrieval. *IEEE Trans Pattern Anal Mach Intell* 2008; 30(1): 89–104.

21. Gionis A, Indyk P and Motwani R. Similarity search in high dimensions via hashing. In: *International conference on very large databases (VLDB)*, 1999, pp.518–529.
22. Andoni A and Indyk P. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In: *IEEE symposium on foundations of computer science (FOCS)*, 2006, pp.459–468.
23. Athitsos V, Potamias M, Papapetrou P, et al. Nearest neighbor retrieval using distance-based hashing. In: *Proceedings of international conference on data engineering (ICDE)*, 2008, pp.327–336.
24. Svonava D and Vlachos M. Visualization graph using minimum spanning dendrograms. In: *Proceedings of the 2010 IEEE international conference on data mining*, 2010, pp.1073–1078.
25. Aggarwal CC, Wolf JL, Wu K-L, et al. Horting hatches an egg: a new graph-theoretic approach to collaborative filtering. In: Proceedings of the fifth ACM SIGKDD international conference on knowledge discovery and data mining, 1999, pp.201–212.
26. Huang Z, Chung W and Chen H. A graph model for e-commerce recommender systems. *JASIST* 2004; 55(3): 259–274.
27. Davidson J, Liebald B, Liu J, et al. The YouTube video recommendation system. In: *Proceedings of the 4th ACM conference on recommender systems (RecSys '10)*, 2010, pp.293–296.
28. Mei T, Yang B, Hua X-S, et al. VideoReach: an online video recommendation system. In: *Proceedings of international conference on information retrieval (SIGIR)*, 2007, pp.767–768.
29. Bennett J and Lanning S. The Netflix prize. In: *Proceedings of KDD cup and workshop*, 2007.
30. Eyjolfsdottir EA, Tilak G and Li N. *MovieGEN: a movie recommendation system*. Technical Report, University of California, Santa Barbara, CA, USA, 2010.
31. Gruvstad F, Gupta N and Agrawal S. *Shiniphy – visual data mining of movie recommendations*. Technical Report, Stanford University, Stanford, CA, USA, 2009.
32. Miller BN, Albert I, Lam SK, et al. MovieLens unplugged: experiences with an occasionally connected recommender system. In: *Proceedings of international conference on intelligent user interfaces (IUI)*, 2003, pp.263–266.
33. O'Donovan J, Smyth B, Gretarsson B, et al. Peerchooser: visual interactive recommendation. In: *Proceeding of the twenty-sixth annual SIGCHI conference on human factors in computing systems*, 2008, pp.1085–1088.
34. Khoshneshin M and Street N. Collaborative filtering via Euclidean embedding. In: *Proceedings of ACM recommender systems*, Barcelona, Spain, September 2010, pp.87–94.
35. Chen K and Liu L. Vista: validating and refining clusters via visualization. *Inform Vis* 2004; 3(4): 257–270,
36. Rheingans P and Landreth C. Principles for effective visualizations. In: Grinstein G and Levkowitz H (eds) *Perceptual issues in visualization*. Berlin: Springer-Verlag, 1995, pp.59–73.
37. Lee R, Slagle J and Blum H. A triangulation method for the sequential mapping of points from N-Space to two-space. *IEEE Trans Comput* 1977; C-26(3): 288–292.
38. Oten R and de Figueiredo R. Topological dimensionality determination and dimensionality reduction based on minimum spanning trees. In: *Proceedings of IEEE international symposium on circuits and systems*, Monterey, CA, 31 May–3 June 1998, pp.366–369.
39. Keim DA, Mansmann F, Schneidewind J, et al. Visual analytics: scope and challenges. In: Simoff SJ, Böhlen MH and Mazeika A (eds) *Visual data mining: theory, techniques and tools for visual analytics*. Berlin Heidelberg: Springer, 2008, pp.76–90.
40. Huang W, Eades P and Hong S-H. Measuring effectiveness of graph visualizations: a cognitive load perspective. *Inform Vis* 2009; 8(3): 139–152,
41. Cerny V. A thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm. *J Opt Theory Appl* 1985; 45: 41–51.
42. Keogh E. Exact indexing of dynamic time warping. In: *Proceedings of 28th international conference on very large data bases*, 2002, pp.406–417.
43. Vlachos M, Kollios G and Gunopulos D. Discovering similar multidimensional trajectories. In: *Proceedings of international conference on data engineering (ICDE)*, 2002, pp.673–684.
44. Chazelle B. A minimum spanning tree algorithm with inverse-ackermann type complexity. *J ACM* 2000; 47: 1028–1047.
45. Karger DR, Klein PN and Tarjan RE. A randomized linear-time algorithm to find minimum spanning trees. *J ACM* 1995; 42(2): 321–328.
46. Lipkus A. A proof of the triangle inequality for the Tanimoto distance. *J Math Chem* 1999; 26: 263–265.
47. Spaeth H. *Cluster analysis algorithms*. Ellis Horwood, 1980.
48. Lee MD and Welsh M. An empirical evaluation of models of text document similarity. In: Bara BG, Barsalou L and Bucciarelli N (eds) *27th Annual meeting of the cognitive science*, 2005, pp.1254–1259.
49. Ando RK. Latent semantic space: iterative scaling improves precision of inter-document similarity measurement. In: *Proceedings of the 23rd annual ACM SIGIR conference on research and development in information retrieval (SIGIR-2000)*, 2000, pp.216–223.