EuroVis Workshop on Visual Analytics (2025)
H.-J. Schulz and A. Vilanova (Editors)

Scalable Force Scheme: a fast method for projecting large datasets

J. Ros[1], A. Arleo[1] and F. Paulovich[1]

[1]Department of Mathematics and Computer Science, Eindhoven University of Technology, The Netherlands

Abstract

Global dimensionality reduction (DR) methods are widely used to project high-dimensional data into a low-dimensional representation, preserving the overall structure of the dataset. Global nonlinear DR techniques allow one to capture complex features of the data but are limited by their high computational cost, making them an unfeasible choice to process large datasets. Force scheme (FS) is one of the most popular of such examples, being adopted in a wide variety of domains, but limiting its application to small datasets. In this paper, we extend FS to improve its convergence quality and speed, by introducing several concepts from gradient descent (GD) theory and lowering its algorithmic complexity. Our new proposed method is less prone to generate distorted projections due to the presence of artifacts, while significantly improving the running times, allowing for nonlinear global DR projections of large datasets.

CCS Concepts

• ***Mathematics of computing*** → ***Dimensionality reduction;***

1. Introduction

In the exploration of multidimensional data, central to many application domains such as data science, machine learning, and natural sciences, it is likely that the problem of representing high-dimensional data will arise [VDMPVDH*09]. Dimensionality reduction (DR) is a discipline that investigates algorithms to reduce the dimensionality of data by projecting its high-dimensional points onto a low-dimensional representation. The resulting projection is meant to retain the topology of the high-dimensional data, so that it remains a reliable proxy to understand its characteristics. Moreover, low-dimensional spaces are much easier to explore: reducing data to a 2D space, for example, makes it possible to use a variety of visualization methods (e.g., scatterplots) to infer the relationships between the data points and the presence of natural clusters, making DR a crucial tool for visual analytics [NA19].

DR is, by definition, a lossy process, meaning that some of the nuances of the original data will be lost and not visible in the final projection. This distortion should be accounted for, and users are required to choose a DR method that best captures the features they are interested in analyzing [EMK*21]. We consider a classification of DR methods on two axes: whether the approach is global or local, and whether it is linear or nonlinear (see Section 2). Following this categorization, *global nonlinear* methods excel at capturing complex features of the overall structure of the dataset, making them very valuable in some analyses [VDMPVDH*09].

In 2003, Tejada et al. [TMN03], in a highly influential paper, introduced a new global nonlinear DR technique, commonly known as *force scheme* (FS). Some years later, Minghim et al. [MPDAL06] further popularized its use with the application of FS to document data with *Interactive Document Map* (IDMAP), and it has been reported to be one of the best DR methods in terms of the quality of the resulting projection [EMK*21].

FS is also popular as a DR technique outside of the visualization community [SSDS*23, BSPN*22]. An example of application of FS in biophysics is the *energy landscape visualization method* (ELViM) [VMS*24], where FS is used to project 3D protein structures onto the 2D plane, approximating their pairwise distances, computed using a specialized distance metric. The main goal of ELViM, as well as other DR methods used with such types of data, is to analyze the overall structure of the complex high-dimensional space, such as the relative size and positioning of clusters, which define protein folding pathways; thus, global nonlinear methods are preferred.

Despite their popularity and effectiveness, global nonlinear methods show limited scalability. This is a consequence of the nature of global DR techniques, which aim to create a projection that approximates distances from the original space, and thus take into account all $N^2/2$ pairwise distances. ELViM prefers FS over other popular global nonlinear methods such as metric multidimensional scaling (mMDS) due to its faster computation time [OFP*14]; however, the DR projection is still too costly to apply to larger datasets, causing researchers to opt instead for fast linear methods, such as principal component analysis (PCA), or local ones, such as uniform manifold approximation and projection (UMAP) [TG19, TWT21]. Moreover, we have found that FS can be prone to generating projection artifacts in the resulting layouts. These can be hard to identify using quantitative metrics, but produce visible distortions in the re-

DOI: 10.2312/eurova.20251098

sulting set of points that can lead uninformed users to make wrong assumptions about the structure of the underlying data.

Within this context, we revisit the original FS algorithm and present *scalable force scheme* (SFS), an extended FS method that includes two major improvements to the original: first, we introduce a gradient descent (GD) framework that improves the quality of its projections and reduces the appearance of artifacts, and second, we accelerate its convergence by greatly decreasing the computational complexity of each iteration. Our paper contributions are as follows: (i) we introduce SFS, explaining its improvements over the original FS; (ii) we conduct an experimental study to show that our system retains the quality of FS while greatly reducing the running time; (iii) we provide examples of projections of large datasets that original FS would not be able to produce in reasonable time.

2. Related work

A DR technique, or projection, is a function $\alpha : \mathbb{R}^D \to \mathbb{R}^q$, where $q \ll D$, intended to produce a low-dimensional representation that retains certain features of the topology of the original high-dimensional space. Typically $q = 2$, allowing the resulting set of points to be visualized as a 2D scatterplot.

DR methods can be divided into *global* or *local*, according to which features of the data they focus on preserving. Global methods consider the whole ensemble of points at once and produce a layout that captures the overall structure as faithfully as possible. Instead, local methods try to preserve detailed neighborhoods around individual points, often taking into account the distances to the k nearest neighbors. Another classification of DR methods is according to the type of transformation of the original data they perform; they are divided into *linear* and *nonlinear*. Linear methods are computationally fast, since the projection is expressed in a single matrix multiplication; however, nonlinear methods allow one to capture much more complex features of the high-dimensional data.

PCA [Pea01], classic multidimensional scaling (MDS) [Tor52], random projection [BM01], and part-linear multidimensional projection (PLMP) [PSN10] are examples of global linear methods; while FS [TMN03], Sammon mapping [Sam69], and mMDS [Kru64] are global nonlinear ones. Local methods can be locally linear, such as locally linear embedding (LLE) [RS00]; however, nonlinear methods are more popular: examples include t-distributed stochastic neighbor embedding (t-SNE) [HR02], UMAP [MHM18], and least square projection (LSP) [PNML08].

3. Methodology

In this section, we first formally define the original FS algorithm, as introduced in [TMN03] and still in use today. We then introduce *gradient force scheme* (GFS), featuring a set of improvements over FS by considering it within a gradient descent (GD) framework and serving as an intermediate step towards SFS, which builds on GFS to make it computationally faster and able to process large datasets.

3.1. FS - The original force scheme

Given a set P of D-dimensional points, FS tries to find a projection $\alpha(p) : \mathbb{R}^D \to \mathbb{R}^q$ that preserves global distances by minimizing:

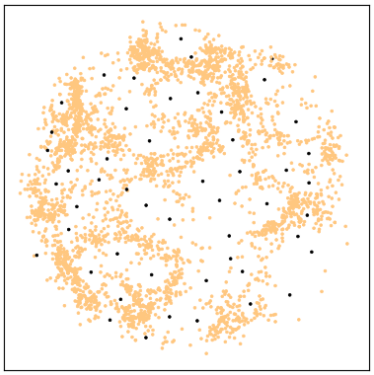

Figure 1: *Projection of the* imdb *dataset using FS. Highlighted in black are the last 50 anchor points in the iteration loop.*

$$\min_{\alpha} \sum_{p_1,p_2 \in P} |d_D(p_1,p_2) - d_q(\alpha(p_1),\alpha(p_2))|, \tag{1}$$

where d_D and d_q are distance criteria (e.g., Euclidean distance) in D- and q-dimensional space, respectively. In practice, optimization is performed given a predefined set of high-dimensional points P, and FS minimizes over their projections $\alpha(p_i)$ directly, rather than explicitly computing the nonlinear function α.

FS starts from an initial projection in q-dimensional space, and iterates several times over all pairs of projected points (p_1, p_2), moving them closer or farther apart according to their distance in the original space. Points are iterated sequentially: given a point $p_1 \in P$, it acts as an *anchor* point by keeping its projection $\alpha(p_1)$ fixed in place, while the projection of every other point $\alpha(p_2), p_2 \neq p_1$ is moved a certain amount Δ in the direction $\vec{v} = \alpha(p_2) - \alpha(p_1)$ to better match the distance between p_1 and p_2 in the projected space with that of the original. This approach, different from common stress majorization algorithms used in other methods such as mMDS, allows FS to converge more rapidly, but also makes it order dependent (i.e., permuting the order of the points in the dataset will produce different results); moreover, every iteration has a cost $O(N^2)$, making FS scale poorly with the size of the data.

Another undesired effect of this minimization procedure is the creation of artifacts in the projection, where the last points to be used as anchors (p_1) before stopping the optimization tend to have a bigger impact on the final layout than other points. This is especially noticeable in some cases, such as the one shown in Figure 1, where one can see individual *island points*, that are isolated from all other points in the dataset. The appearance of these artifacts depends on a variety of factors, such as the dataset itself, the distance metric, or the hyperparameters used. They are also hard to detect quantitatively, since it is only a small fraction of points that are severely distorted, causing common projection quality metrics to not penalize it enough (see Section 4); however, they greatly distort the final appearance of the layout, which is crucial for global visual analyses such as those typically done with FS, and can cause the end user to derive erroneous conclusions from the visualization.

3.2. GFS - Introducing a gradient descent framework

From the explanation of the original FS algorithm above, two implementation questions become evident: (1) how much should points be moved at each iteration? And (2) how many iterations should be performed before reaching the final layout? The original FS paper [TMN03] did not explicitly elaborate on these two points, leaving it to the user to make a decision based on the data set and computational limitations. However, setting the correct values is crucial to obtain a good result, and they can be hard to guess, especially during the exploration of a dataset when one does not know at all how the projection should look like.

GFS is inspired by gradient descent (GD), an active field of research in recent decades, especially after the surge in popularity of neural networks [Rud17]. GD algorithms are designed to optimize a differentiable function following a variety of strategies; in particular, stochastic gradient descent (SGD) is commonly used to speed up convergence by using approximations of the gradient.

Although the original formulation of the problem, as presented in Equation 1, is not differentiable due to the presence of the absolute value and thus cannot be solved with GD directly, there exist several similarities in the formulation of the problem that allow us to use GD concepts in the optimization process of FS. Firstly, we apply the notion of **learning rate decay** to control the amount of movement to apply to individual points at each iteration, which should decrease with time, allowing the layout to converge to a local minimum. Thus, we explicitly consider hyperparameters for the *initial learning rate* $\eta_0 > 0$ and *decay* $\gamma \in (0,1]$. At each iteration, the learning rate is updated with $\eta_{t+1} = \eta_t \cdot \gamma$, where η_t denotes the value of the learning rate in iteration t. This allows for a smooth exponential decay of the learning rate.

Secondly, we introduced a **convergence criterion** that allows FS to stop early if the layout is not improving across iterations, indicating that the process has reached a local minimum. To this end, we use an error metric introduced in [TMN03] and defined as:

$$\varepsilon = \frac{1}{N^2} \sum_{p_1,p_2 \in P} |d_D(p_1,p_2) - d_q(\alpha(p_1),\alpha(p_2))|, \qquad (2)$$

where N is the number of points in P. Note that this corresponds to a scaling (i.e., the average) of the objective function that FS tries to minimize, as defined in Equation 1. Since the magnitude of ε depends on the dataset, it was originally used to compare different DR projections of the same data; however, it can also be used to measure the evolution of the projection error during optimization.

We define a *tolerance* τ hyperparameter to early-stop the optimization if the error is not decreasing sufficiently enough. Since the optimization process can be noisy and not always monotonically decreasing, we use an *error window* $[\varepsilon_{t-W},\ldots,\varepsilon_{t-1}]$ of length W, and consider the optimization to have converged if and only if the current error is higher than the average of the last W iterations. Additionally, a maximum number of iterations can still be specified to stop the optimization if it has not yet converged.

Finally, we propose to **iterate over all points in a random order**, different at each iteration. It is common practice in SGD to randomly permute the dataset to break possible biases in the order of the samples, such as datasets that are ordered according to some attribute or cluster. Note that to ensure that FS is still *deterministic* (i.e., runs with the same data and hyperparameters produce the exact same result), we need a fixed seed for the random orderings, which was already done when choosing a random initial projection of points. However, this makes FS still *order-dependent* (i.e. different orderings of the same dataset will produce different results).

3.3. SFS - Scalable force scheme

Global nonlinear DR methods usually struggle with large datasets, since they need to consider distances between all pairs of points, as opposed to local methods, which generally only focus on neighborhoods around each point. FS can be preferred over alternatives such as mMDS due to its fast convergence speed, but nonetheless it still scales poorly when used on larger datasets, limiting its usability.

We consider a series of modifications to the previously described GFS to accelerate its convergence with minimal impact on the quality of the layout. Our intuition to speed-up the procecss is that, to approximate the optimal layout starting from a random positioning, one does not need to compute all $N^2/2$ pairwise distances at each iteration; in order to estimate the direction in which a point needs to be displaced in q-dimensional space, it is enough to compute a relevant yet reduced subsample of distances. Our new approach considers **moving each point based on a random subsample** of $\sqrt{N}$ points, P'. This reduces the algorithmic complexity of each iteration from $O(N^2)$ to $O(N\sqrt{N})$, vastly improving the scalability to large datasets. Our implementation considers, for each iteration, every point $p_1 \in P'$ and moves all other N points $p_2 \in P$ accordingly; the choice of subsampling the set of anchors p_1 instead of moved points p_2 is an implementation detail, as it allows for more efficient parallelization of each iteration.

Although not explicitly discussed in [TMN03], implementations of the original FS technique [Pau22] often rely on the precomputation of the matrix of pairwise distances of points in D-dimensional space. This avoids having to compute all N^2 D-dimensional distances repeatedly, speeding up each iteration. However, the cost of storing the matrix in memory becomes prohibitive when using large datasets. SFS reduces the total number of pairwise distance calculations, making the **precomputation of a square distance matrix redundant**, even for small datasets (see Section 4).

The original FS algorithm as defined in [TMN03] requires normalization of the set of distances at every step. This is performed only once at the beginning for the original data, since D-dimensional distances remain constant, but it must be repeated for the 2-dimensional projection. The authors argued that normalizing the coordinates of the projected points is a much faster alternative ($O(N)$) than computing the square distance matrix at every iteration ($O(N^2)$), which they still do once initially for the high-dimensional space. In SFS, in order to avoid the computation of the pairwise distance matrix, we **removed the normalization step** altogether, since we found it unnecessary with reasonable values of η_0 and γ. This also avoids the need of de-normalizing the distances at the end of the process. The pseudocode for SFS is detailed in Algorithm 1.

Algorithm 1 Scalable force scheme (SFS)

Input:

P_D	Set of N D-dimensional points
$Iter_{max}$	Maximum number of iterations
η_0	Initial learning rate
γ	Decay factor
τ	Error tolerance
W	Error window size

Output:

P_q	q-dimensional coordinates of the N points

$P_q \leftarrow$ Random positioning of N points in $[0,1]^q$
$\eta \leftarrow \eta_0$
for $i = 1$ **to** $Iter_{max}$ **do**
  $\eta \leftarrow \eta \cdot \gamma$
  $Idx \leftarrow$ Random permutation of size N
  $Idx_{anchor} \leftarrow$ random subsample of $\sqrt{N}$ indices
  $\varepsilon_i \leftarrow 0$
  for each $p_1 \in Idx_{anchor}$ **do**
    for each $p_2 \in Idx, p_2 \neq p_1$ **do**
      $\vec{v} \leftarrow P_q[p_2] - P_q[p_1]$
      $d_q \leftarrow ||P_q[p_2] - P_q[p_1]||$
      $d_D \leftarrow ||P_D[p_2] - P_D[p_1]||$
      $\Delta \leftarrow d_D - d_q$
      $P_q[p_2] \leftarrow P_q[p_2] + \eta \cdot \Delta \cdot \vec{v}/d_q$
      $\varepsilon_i \leftarrow \varepsilon_i + |\Delta|$
    end for
  end for
  if $i > W$ **and** $\frac{1}{W}\sum_{j=i-1}^{i-W}(\varepsilon_j) - \varepsilon_i < \tau$ **then**
    break
  end if
end for
Return P_q

4. Results

In this section, we discuss the quantitative and qualitative results of our proposed algorithm, SFS, in comparison with the original FS and GFS, presented in Section 3. In the first part, we conduct an experimental evaluation on the datasets from the Espadoto et al. survey [EMK*21] to quantitatively compare GFS and SFS to the original FS. As mentioned in Section 3, quantitative metrics, such as stress, cannot capture the distortion of the layout produced by island points, leading us to conduct a qualitative evaluation by showing the projection of some example datasets for visual comparison. Finally, we demonstrate the scalability of SFS, when used to process very large datasets, which current global nonlinear DR methods are not able to accomplish in a reasonable amount of time.

Our implementation of FS, GFS and SFS, as well as the code used to generate all the results presented in this section, are publicly available [RAP]. The experiments discussed below were conducted on a laptop featuring a 13th Gen Intel® Core™ i7-13700H processor, 16 GiB of memory, and running Ubuntu 24.04.

Quantitative Comparison. We measure the quality of a projected layout, with respect to the original set of distances in high-dimensional space, using stress as defined in [EMK*21]:

$$\frac{\sum_{i<j}(d_D(p_i,p_j) - d_q(\alpha(p_i),\alpha(p_j)))}{\sum_{i<j} d_D(p_j,p_i)}. \quad (3)$$

We do not report the comparison using other local quality metrics, such as continuity and trustworthiness, as they capture the goodness of a projection on a local scale, which is not the intended use of global methods such as FS (see Section 2); they can, however, be found in the supplementary material.

Dataset	N	D	FS	GFS	SFS
bank	2059	63	11.13	9.62	0.44
cifar10	3250	1024	29.08	30.91	1.34
cnae9	1080	856	3.96	3.47	0.28
coil20	1440	400	6.12	4.67	0.16
epileptic	5750	178	73.16	68.9	0.53
fashion_mnist	3000	784	23.54	22.85	0.61
fmd	997	1536	3.66	3.87	0.16
har	735	561	1.97	2.04	0.08
hatespeech	3221	100	23.38	20.04	0.12
hiva	3076	1617	30.13	27.18	0.62
imdb	3250	700	27.72	24.13	0.47
orl	400	396	1.13	0.76	0.11
secom	1567	590	7.65	6.24	0.33
seismic	646	24	2.82	1.75	0.03
sentiment	2748	200	18.36	16.12	0.15
sms	835	500	2.32	1.98	0.09
spambase	4601	57	45.71	44.76	0.15
svhn	732	1024	2.15	2.82	0.16

Table 1: *Execution time, in seconds, of FS, GFS, and SFS on the 18 example datasets from [EMK*21]. N is for the number of data points and D the number of dimensions. The reported value (in seconds) is the average over 10 iterations.*

For each dataset, we compute 10 different projections, using different random seeds to control the initial placement of points and the order in which they are iterated (when relevant). The values for hyperparameters were chosen through an empirical exploration of the space and are kept the same for all datasets and repetitions; they are available in the supplementary material. In Table 1 we show both the size of the datasets we used for this first comparison and the average time spent per projection. Figure 2 shows the stress metric computed after the projection of each dataset.

Concerning running times, GFS is generally slightly faster than FS, due to the early-stopping criterion, which causes FS to make unnecessary iterations that do not improve the layout; however, SFS is approximately 2 orders of magnitude faster than the other two, especially in those datasets where $N \gg D$. In terms of stress, the difference in quality of the three methods is very minor. This is especially relevant for SFS, since it shows that it is able to reach levels of quality comparable to those of FS with significantly reduced running times.

Qualitative Comparison Figure 3 compares the projection of the *imdb* dataset, found to cause FS to generate artifacts and island points (see Section 3), using FS and GFS. In the GFS projection, no

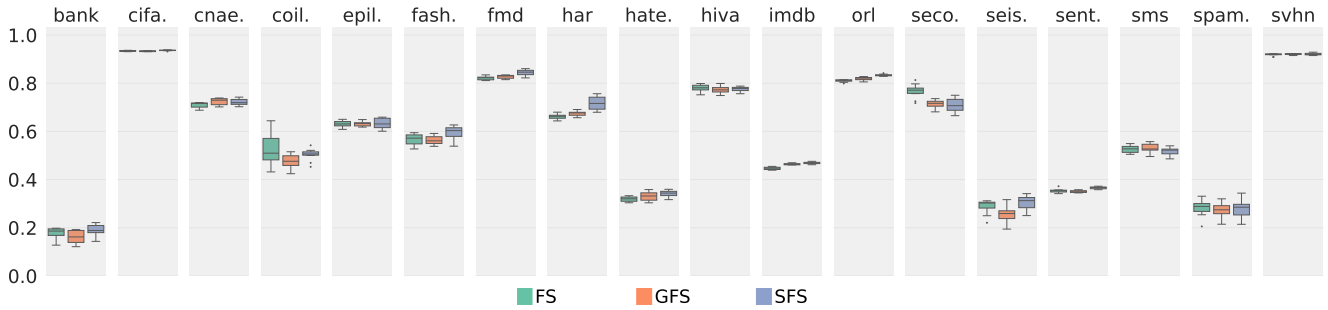

Figure 2: *Boxplots showing the quality of the resulting layouts, measured with stress, where lower is better. Each column represents a different dataset. For every dataset, colors green, red, and blue represent the different setups: FS, GFS, and SFS, respectively.*

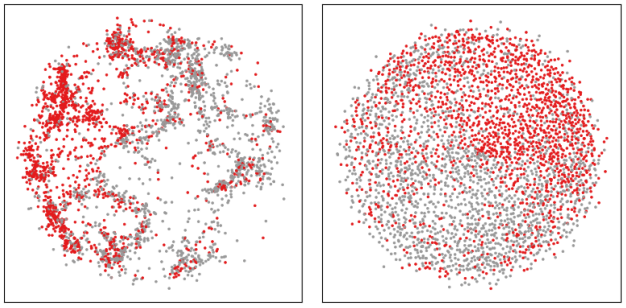

Figure 3: *Projection of the* imdb *dataset using FS (left) and GFS (right). Points are colored according to their binary class, as defined in the original data.*

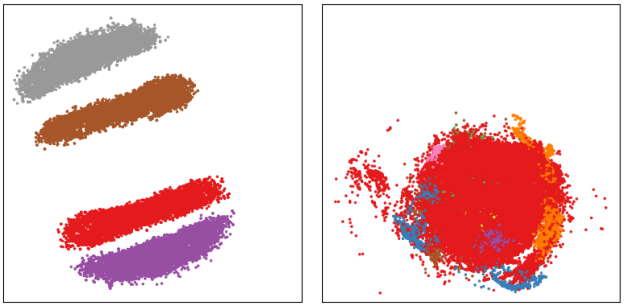

Figure 4: *Projection of the* Quadruped Mammals *[LF89] (left) and* Fibers *[PEPM12] (right) datasets, with* $N = 50k, D = 71$ *and* $N = 250k, D = 30$ *respectively.*

clusters are visible, although globally class separation can still be observed. Comparing this result with the layout obtained with FS, it can be seen that artifacts created by FS can generate a very poor global layout, leading to erroneous interpretations of the data, such as a false sense of structure. The correctness of the global layout produced by GFS can be further verified by running FS for many iterations with a very low learning rate, resulting in a projection similar to that of GFS.

In some cases, the layouts produced by SFS can be qualitatively worse than GFS (a compromise to be made by the user in exchange of fast computation time) but are still generally better than FS. All projections by the different setups for the datasets of Table 1 are available as supplementary material.

Finally, in Figure 4 we show an example of the use of SFS to project very large datasets, where other global nonlinear methods, including FS and GFS, are not a feasible option due to their high computational times. For the two datasets shown in Figure 4, GFS took 97 and 1805 seconds, respectively, to perform *a single* iteration, while SFS was able to reach the final projection (after 16 and 23 iterations, respectively) in 9 and 90 seconds in total. Note that for large datasets, the pairwise distance matrix cannot be precomputed in FS and GFS due to memory limitations, causing iterations to take longer to execute.

5. Conclusions

In this paper, we have revisited the FS algorithm for DR. We have identified limitations affecting the quality of the layouts and the scalability to large datasets, a common problem among global nonlinear DR methods. We have introduced SFS, a global nonlinear DR method that is able to produce high-quality projections of large datasets. SFS builds on top of FS in two steps: first, we improve the optimization process of FS by introducing GD concepts, leading to an intermediate algorithm: GFS; second, we extend GFS to reduce the number of steps needed to converge, greatly reducing the computation time with a limited impact on the projection quality. We complement the previous contributions with experimental results that measure the quality of the projections. We show that, quantitatively, GFS and SFS are similar to the original FS; however, GFS is less prone to generate critical artifacts, present in some cases with FS, and SFS further improves the method by significantly reducing the computation time with little effect on the projection stress.

Future work can conduct a more extensive exploration of the hyper-parameter space, fine-tuning the default SFS, to ensure that the layout quality is as good as GFS while still maintaining reduced running times. Moreover, we show the need for a DR quality metric that is able to capture the distortions produced by island points, which would allow detecting the presence of such, and possibly other, artifacts.

References

[BM01] BINGHAM E., MANNILA H.: Random projection in dimensionality reduction: applications to image and text data. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2001), KDD '01, Association for Computing Machinery, p. 245–250. doi:10.1145/502512.502546. 2

[BSPN*22] BONDANCIA T. J., SOARES A. C., POPOLIN-NETO M., GOMES N. O., RAYMUNDO-PEREIRA P. A., BARUD H. S., MACHADO S. A., RIBEIRO S. J., MELENDEZ M. E., CARVALHO A. L., REIS R. M., PAULOVICH F. V., OLIVEIRA O. N.: Low-cost bacterial nanocellulose-based interdigitated biosensor to detect the p53 cancer biomarker. *Biomaterials Advances 134* (Mar. 2022), 112676. doi:10.1016/j.msec.2022.112676. 1

[EMK*21] ESPADOTO M., MARTINS R. M., KERREN A., HIRATA N. S. T., TELEA A. C.: Toward a quantitative survey of dimension reduction techniques. 2153–2173. doi:10.1109/TVCG.2019.2944182. 1, 4

[HR02] HINTON G. E., ROWEIS S.: Stochastic neighbor embedding. *Advances in neural information processing systems 15* (2002). 2

[Kru64] KRUSKAL J. B.: Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika 29*, 1 (1964), 1–27. doi:10.1007/BF02289565. 2

[LF89] LANGLEY P., FISHER D.: Quadruped Mammals. UCI Machine Learning Repository, 1989. DOI: https://doi.org/10.24432/C5860F. 5

[MHM18] MCINNES L., HEALY J., MELVILLE J.: Umap: Uniform manifold approximation and projection for dimension reduction. URL: https://arxiv.org/abs/1802.03426, doi:10.48550/ARXIV.1802.03426. 2

[MPDAL06] MINGHIM R., PAULOVICH F. V., DE ANDRADE LOPES A.: Content-based text mapping using multi-dimensional projections for exploration of document collections. Erbacher R. F., Roberts J. C., Gröhn M. T., Börner K., (Eds.), p. 60600S. URL: http://proceedings.spiedigitallibrary.org/proceeding.aspx?doi=10.1117/12.650880, doi:10.1117/12.650880. 1

[NA19] NONATO L. G., AUPETIT M.: Multidimensional projection for visual analytics: Linking techniques with distortions, tasks, and layout enrichment. *IEEE Transactions on Visualization and Computer Graphics 25*, 8 (Aug. 2019), 2650–2673. doi:10.1109/TVCG.2018.2846735. 1

[OFP*14] OLIVEIRA A. B., FATORE F. M., PAULOVICH F. V., OLIVEIRA O. N., LEITE V. B. P.: Visualization of protein folding funnels in lattice models. *PLoS ONE 9*, 7 (July 2014), e100861. doi:10.1371/journal.pone.0100861. 1

[Pau22] PAULOVICH F.: Forcescheme code. https://github.com/fpaulovich/dimensionality-reduction/tree/main/force, 2022. 3

[Pea01] PEARSON K.: Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science 2*, 11 (1901), 559–572. 2

[PEPM12] POCO J., ELER D. M., PAULOVICH F. V., MINGHIM R.: Employing 2d projections for fast visual exploration of large fiber tracking data. *Computer Graphics Forum 31*, 3pt2 (June 2012), 1075–1084. doi:10.1111/j.1467-8659.2012.03100.x. 5

[PNML08] PAULOVICH F. V., NONATO L. G., MINGHIM R., LEVKOWITZ H.: Least square projection: A fast high-precision multidimensional projection technique and its application to document mapping. *IEEE Transactions on Visualization and Computer Graphics 14*, 3 (2008), 564–575. doi:10.1109/TVCG.2007.70443. 2

[PSN10] PAULOVICH F. V., SILVA C. T., NONATO L. G.: Two-phase mapping for projecting massive data sets. *IEEE Transactions on Visualization and Computer Graphics 16*, 6 (2010), 1281–1290. doi:10.1109/TVCG.2010.207. 2

[RAP] ROS J., ARLEO A., PAULOVICH F.: Gradient/scalable force scheme source code. https://github.com/migamic/dimensionality-reduction. 4

[RS00] ROWEIS S. T., SAUL L. K.: Nonlinear dimensionality reduction by locally linear embedding. *Science 290*, 5500 (2000), 2323–2326. doi:10.1126/science.290.5500.2323. 2

[Rud17] RUDER S.: An overview of gradient descent optimization algorithms, 2017. doi:10.48550/arXiv.1609.04747. 3

[Sam69] SAMMON J.: A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers C-18*, 5 (1969), 401–409. doi:10.1109/T-C.1969.222678. 2

[SSDS*23] SOARES A. C., SOARES J. C., DOS SANTOS D. M., MIGLIORINI F. L., POPOLIN-NETO M., DOS SANTOS CINELLI PINTO D., CARVALHO W. A., BRANDÃO H. M., PAULOVICH F. V., CORREA D. S., OLIVEIRA O. N., MATTOSO L. H. C.: Nanoarchitectonic e-tongue of electrospun zein/curcumin carbon dots for detecting staphylococcus aureus in milk. *ACS Omega 8*, 15 (Apr. 2023), 13721–13732. doi:10.1021/acsomega.2c07944. 1

[TG19] TRIBELLO G. A., GASPAROTTO P.: Using dimensionality reduction to analyze protein trajectories. *Frontiers in Molecular Biosciences 6* (June 2019), 46. doi:10.3389/fmolb.2019.00046. 1

[TMN03] TEJADA E., MINGHIM R., NONATO L. G.: On improved projection techniques to support visual exploration of multi-dimensional data sets. 218–231. doi:10.1057/palgrave.ivs.9500054. 1, 2, 3

[Tor52] TORGERSON W. S.: Multidimensional scaling: I. theory and method. *Psychometrika 17*, 4 (1952), 401–419. 2

[TWT21] TROZZI F., WANG X., TAO P.: Umap as a dimensionality reduction tool for molecular dynamics simulations of biomacromolecules: A comparison study. *The Journal of Physical Chemistry B 125*, 19 (May 2021), 5022–5034. 1

[VDMPVDH*09] VAN DER MAATEN L., POSTMA E. O., VAN DEN HERIK H. J., ET AL.: Dimensionality reduction: A comparative review. *Journal of machine learning research 10*, 66-71 (2009), 13. 1

[VMS*24] VIEGAS R. G., MARTINS I. B. S., SANCHES M. N., OLIVEIRA JUNIOR A. B., CAMARGO J. B. D., PAULOVICH F. V., LEITE V. B. P.: Elvim: Exploring biomolecular energy landscapes through multidimensional visualization. *Journal of Chemical Information and Modeling 64*, 8 (Apr. 2024), 3443–3450. doi:10.1021/acs.jcim.4c00034. 1