International Journal of Data Science and Analytics
https://doi.org/10.1007/s41060-022-00348-7

REGULAR PAPER

Optimizing graph layout by t-SNE perplexity estimation

Chun Xiao[1] · **Seokhee Hong**[2] · **Weidong Huang**[1]

Received: 31 March 2022 / Accepted: 3 July 2022

Abstract
Perplexity is one of the key parameters of dimensionality reduction algorithm of t-distributed stochastic neighbor embedding (t-SNE). In this paper, we investigated the relationship of t-SNE perplexity and graph layout evaluation metrics including graph stress, preserved neighborhood information and visual inspection. As we found that a small perplexity is correlated with a relative higher normalized stress while preserving neighborhood information with a higher precision but less global structure information, we proposed our method to estimate appropriate perplexity either based on a modified standard t-SNE or the sklearn Barnes–Hut TSNE. Experimental results demonstrate effectiveness and ease of use of our approach when tested on a set of benchmark datasets.

Keywords Data visualization · Dimensionality reduction · Graph layout · Graph/network data · Perplexity · t-SNE

1 Introduction

A particular class of graph layout methods is based on dimensionality reduction (DR) techniques, which are designed to embed original high-dimensional data into a two or three-dimensional data, so that the graph is visually readable [1,2]. Some examples of DR techniques successfully applied for graph layout include linear DR such as principal component analysis (PCA) [3], sampling-based approximation DR [4,5], nonlinear DR such as t-SNE [6–8], where the graph-theoretic distances between pairs of nodes in a graph are mapped as the original high-dimensional data for DR [3,6,9,10]. The aim of the DR techniques based on the graph-theoretic distance is to minimize the difference between the graph-theoretic distance and the two-dimensional Euclidean distance, such that the two-dimensional layout reflects as much as possible the pair-wiser high-dimensional distances within a given neighborhood.

t-SNE [11] is one of the nonlinear DR algorithms which has been widely used in various applications. It is also one of the most successful DR algorithms for graph layouts including IDMAP, UMAP and PBC [12]. Among the set of parameters which determine the t-SNE performance, perplexity is the one showing randomness, as different perplexities often yield different visualizations. Maaten and Hinton [11] suggested typical values of the perplexity ranging between 5 and 50. Later, Wattenberg et al. [13] demonstrated examples with various settings of perplexity, which lead to a variety of shapes, clusters and topology of the two-dimensional visualization, and they claimed that "getting the most from t-SNE may mean analyzing multiple plots with different perplexities." It seems to be the reality that in practices, the perplexity is usually obtained by a random initialization, or a guess based on experiences, or a grid search, the one with the best performance out of the tries is then selected. For large datasets, a grid search is extremely time-consuming, which is not a good practice.

The question about how to estimate an appropriate t-SNE perplexity is still left open, although it is indicated that the most appropriate value of perplexity depends on the density of data, which means that a larger and/or denser dataset requires a larger perplexity [14]. We are also very interested in how to estimate a perplexity which is appropriate to gener-

Seokhee Hong and Weidong Huang both authors contributed equally to this study.

✉ Chun Xiao
chun.xiao@uts.edu.au

Seokhee Hong
Seokhee.Hong@sydney.edu.au

Weidong Huang
weidong.huang@uts.edu.au

[1] University of Technology Sydney, 15 Broadway, Ultimo, NSW 2006, Australia

[2] University of Sydney, 1 Cleveland St, Camperdown, NSW 2007, Australia

Published online: 30 July 2022

ate graph layouts with good quality and less random output, and is easy to use without requirement to try multiple times. Since the perplexity is closely related to the size and density of the data as suggested above, we explore the following questions to reveal the truth behind the *mysterious* perplexity of t-SNE in this paper. The questions include:

1. What it the relationship between different perplexities and the size of a graph dataset?
2. Is there a reasonable perplexity range for a good-quality graph layout? If the answer is yes, what role the size and density of a graph plays when estimating a reasonable perplexity?

In the remaining of this paper, we first review the state-of-the-art tools employing t-SNE to draw graphs. We then design our approach to investigate t-SNE perplexity. The experimental results are finally presented to validate our approach. Our contributions are:

1. We modify the standard t-SNE [14] to fit underestimated perplexity as a valid input to improve its usability (the standard t-SNE does not accept the underestimated perplexity as an valid input).
2. We explore the impact of different perplexities on graph layout, identify the relationship between the perplexity and graph layout quality, and propose a perplexity estimation approach for good graph layouts using the standard t-SNE.
3. We further propose an adapted version of perplexity estimation which can be applied with sklearn Barnes–Hut TSNE [15] to accelerate the process while keeping good layout quality for large graph datasets.

Tested on a set of benchmark datasets, our proposed approach demonstrates better performance when compared with one of the most successful methods tsNET* [6], by reducing 5% of the normalized stress and increasing 2.5% of the neighborhood preservation on average. With our adapted version for perplexity estimation of sklearn Barnes–Hut TSNE [15], the runtime on large datasets can be reduced from hundreds of seconds with the standard t-SNE to tens of seconds, meanwhile keeping comparable even better normalized stress and neighborhood preservation. Our proposed approach presents advantages with respect to effectiveness and ease of use.

2 Related work

In this section, we review the related work which employed t-SNE to draw graph, and introduce the definition of perplexity in t-SNE and metrics for graph layout evaluation applied in this paper.

2.1 Graph drawing based on t-SNE and t-SNE variations

Kruiger et al. [6] demonstrated that their employment of t-SNE, named tsNET, outperforms several other DR algorithms like IDMAP in graph layout, where tsNET simplifies the input parameter settings to focus on perplexity only with minimal tuning of other parameters such as learning rate and number of iterations. Based on tsNET, tsNET* initializes a layout using PivotMDS, and optimizes the layout based on the PivotMDS layout. It is mentioned that the perplexities were set on average of 80 with a standard deviation of 45 for most of the test datasets. However, it has not been discussed how perplexity for each dataset was selected (except EVA dataset) and if different perplexity affects the graph layout, tsNET rejects small perplexity as input.

Xu et al. [7] drew graphs using a model named t-NeRV [16], which was a generalized t-SNE model based on a modified graph-theoretic distance matrix, in which the cost function is a combination of the average of both of the Kullback–Leibler divergence from high-dimensional space to low-dimensional space and vice versa, the cost of one of the energy model out of a modified Kamada and Kawai (KK) model, a modified Fruchterman-Reingold (FR) model, and a modified Linlog model. Their approach also showed that the output graph layout is quite dependent on the fine-tuning of a set of parameters controlling early compression, adjacent neighbor compression, node repulsive forces, and edge attractive forces during the cost computation. Again, how the perplexity was selected for different results presented in the paper has not been mentioned.

In addition to drawing graph based on the graph-theoretic distances, a parametric version of t-SNE named GraphTSNE [8] adopted additional node features in the final graph layout. GraphTSNE trains a two-layer residue gated graph convolutional network using a modified t-SNE cost C_T consisting of both the graph clustering loss C_G and the feature clustering loss C_X, so that $C_T = \alpha C_G + (1-\alpha)C_X$. The graph clustering loss is computed based on the shortest graph-theoretic distance, while the feature clustering loss is computed based on the Euclidean distance of word embedding of each node. An important learning goal is to optimize the trade-off weight α for the two loss components, and the perplexity is set to 30. However, when tested with tsNET, the CORA dataset works when the perplexity is set bigger than 169. The impact of different perplexities in GraphTSNE has not been investigated.

We can observe that the perplexity initialization of the use cases mentioned above shows randomness. As a fact, optimization of perplexity has raised attention of some researchers. De Rosa et al. [17] investigated several perplex-

ity meta-heuristic optimization methods including artificial bee colony algorithm, bat algorithm, genetic programming, and particle swarm optimization on word embedding visualization. A practical issue is that the optimization methods are evaluated by the Kullback–Leibler divergence cost residue, which is closely related to perplexity itself. That means different perplexities correspond to different cost when initializing, the cost residues for different perplexities are not comparable. De Bodt et al. [18,19] tried perplexity-free parametric t-SNE, and also tried to adjust the embeddings (visualization) by using additional class labels instead of the traditional perplexity [20]. Their methods introduced supervised training process. As in our case, t-SNE will be employed for unsupervised DR, we will first investigate the impact of different perplexities on graph layout quality, then test our approach to estimate appropriate value of this parameter on benchmark datasets.

2.2 Definition of perplexity in t-SNE

The perplexity in t-SNE is 2 to the power of Shannon entropy of the conditional distribution induced by a data point x_i (see Eq. 1). As explained by Maaten [14], "the perplexity of a fair die with k sides is equal to k. In t-SNE, the perplexity may be viewed as a knob that sets the number of effective nearest neighbors. It is comparable with the number of nearest neighbors k that is employed in many manifold learners."

$$\mathrm{Perp}(P_i) = 2^{H(P_i)} \tag{1}$$

where

$$H(P_i) = -\sum_j p_{j|i} \log_2 p_{j|i} \tag{2}$$

and

$$p_{j|i} = \frac{\exp(-\left\|x_i - x_j\right\|^2 / 2\sigma_i^2)}{\sum_{k(k\neq i)} \exp(-\left\|x_i - x_k\right\|^2 / 2\sigma_i^2)}, \quad p_{i|i} = 0 \tag{3}$$

where $p_{j|i}$ is the conditional probability of data point x_j in the neighborhood of x_i based on a Gaussian distribution. t-SNE defines joint probabilities p_{ij} by symmetrizing the two conditional probabilities as follows:

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2}. \tag{4}$$

Note that the bandwidth of the Gaussian kernels, i.e., σ_i in Eq. 3, is set in a way that $H(P_i)$ equals a predefined perplexity [15]. In addition, σ_i is different for each data point x_i, which means that it changes for each high-dimensional instance in order to capture the variations in density for different high-dimensional neighborhoods, where σ_i tends to be smaller in regions of the data space with a higher data density compared to the regions with lower density [15]. In practices, σ_i is found iteratively by trial-and-error [21], using binary search, until a user-defined perplexity H is reached. With this process the probability matrix P in the high-dimensional space is obtained. Similarly, the probability matrix Q in the low-dimensional space based on t-distribution is computed as

$$q_{ij} = q_{ji} = \frac{(1 + \left\|y_i - y_j\right\|^2)^{-1}}{\sum_{k\neq l}(1 + \|y_k - y_l\|^2)^{-1}}, \; q_{i|i} = 0 \tag{5}$$

Then, Kullback–Leibler divergence C_{KL} as cost is computed to minimize the difference between P and Q, usually by gradient descent.

$$C_{KL} = \sum_{i\neq j} p_{ij} log \frac{p_{ij}}{q_{ij}}. \tag{6}$$

With an underestimated perplexity or the unfounded Gaussian kernel after limited iterations, the data points in the defined neighborhood fail to fit to a Gaussian, resulting in exceptions with undefined $p_{j|i}$. We discuss our solution to this issue in Sect. 3.1.

2.3 Metrics for graph layout evaluation

Espadoto et al. [12] summarized most widely used metrics for graph layout evaluations. Quality metrics include scalar metrics (trustworthiness, continuity, normalized stress, neighborhood hit, visual separation metrics), local metrics (projection precision score, stretching and compression, average local error), and point-pair metrics. As we would like to keep in line with the work of tsNET and tsNET* [6], the graph layout in this paper is evaluated exactly the same as [6] by (1) normalized stress measurement M_σ, (2) neighborhood preservation measurement M_{np}, and (3) visual inspection. Below, we list the formal definitions of normalized stress and neighborhood preservation in [6]. As the visual inspection is not a quantitative metric, we are not able to define it as a metric similar to the other two measurements. Instead, we evaluate a visualization by inspecting the structural distortion or overlapping on the benchmark datasets.

2.3.1 Normalized stress measure M_σ

A key to the DR-approach is to suitably define the distance d so that the stress is minimized to reflect a good layout.

Given a set of N n-dimensional observations $\{x_i \in R^n\}_{i=1}^N$, a DR technique maps x_i to a lower-dimensional set $\{y_i \in$

$R^m\}_{i=1}^N$, where m is usually 2 or 3. The measurement of normalized stress M_σ is computed as:

$$M_\sigma = \sum_{i,j} \left(\frac{\mathrm{d}(x_i, x_j) - \|y_i - y_j\|}{\mathrm{d}(x_i, x_j)} \right)^2 \tag{7}$$

where $d(x_i, x_j)$ is a distance metric over the input space, in this case, is the shortest graph-theoretical distance, and $\|y_i - y_j\|$ usually refers to the Euclidean 2D distance.

2.3.2 Neighborhood preservation measure M_{np}

Given a graph $G = (V, E)$, let $N_G(x_i, r_G) = \{x_j \in V | d_{ij} <= r_G\}$ denote the nodes with a graph-theoretic distance no more than r_G from node x_i. Then, in the low-dimensional space, which is usually the two-dimensional projection space, an equally sized neighborhood of x_i in the layout $N_Y(x_i, k_i)$ is defined as the set of nodes corresponding to the data points that are the k_i -nearest-neighbors of y_i, with $k_i = |N_G(x_i, r_G)|$. Note that k_i may differ for different x_i. The neighborhood preservation measurement M_{np} is then defined as the Jaccard similarity between the neighborhoods in the high- and low-dimensional spaces N_G and N_Y, averaged over G.

$$M_{np} = \frac{1}{|V|} \sum_i \frac{|N_G(x_i, r_G) \cap N_Y(x_i, k_i)|}{|N_G(x_i, r_G) \cup N_Y(x_i, k_i)|} \tag{8}$$

In our tests, we let $r_G = 2$ to compare with the results in the related references.

3 Design of our approach

3.1 Modification on perplexity and bandwidth fitting in standard t-SNE

We make modification based on the Python code available from t-SNE Github [14] to improve handling of exceptions during bandwidth fitting.

Handling exceptions may vary in practices. The original standard t-SNE code based on which we have been working does not accept underestimated perplexities, neither does tsNET. For example, tsNET simply throws the exceptions in case an underestimated perplexity is fed, an increased user-defined perplexity is then expected to try again until it works. It is a reasonable solution in practice, as an increased perplexity means considering a broader neighborhood with more samples for more reliable statistical results. However, the randomness in this perplexity initialization procedure seems to be quite user-unfriendly, especially for some extreme examples like the EVA dataset, which needs almost 600 as the perplexity to start off in tsNET [6].

As we identify that the exceptions are caused by data sparsity, we apply two strategies as solutions to the problem. One strategy is to increase the Gaussian kernel σ_i by a controlled exaggeration rate during the process to find an appropriate β_i, where β_i is a function of σ_i, $\beta_i = \frac{1}{2\sigma_i^2}$, in order to avoid failures when fitting the data points. The exaggeration rate is 2 in our experiment.

Another strategy is to run an intrinsic grid search within limited steps for a bigger perplexity. When detecting an exception during fitting without updating the initial exaggeration rate of Gaussian kernel, the perplexity is increased by a pre-defined rate, which is currently 3 in our experiment. The modifications are described in algorithm 1.

Algorithm 1 Modification on bandwidth and perplexity fitting in the standard t-SNE

Require: $distance\ matrix\ D$
Ensure: probability matrix P without undefined value.
1: **if** $exp(-d_{ik}^2 * \beta_i), k \neq i$, is undefined, where d_{ik} is the (i_{row}, k_{column}) cell in D **then**
2: **if** strategy 1: increase σ_i **then**
3: **while** number of tries < pre-defined number 1 **do**
4: β_i / = exaggerate rate
5: recompute p_{ij}
6: **end while**
7: **end if**
8: **if** strategy_2: increase user-defined perplexity **then**
9: **while** $number\_of\_tries < pre\_defined\_number\_2$ **do**
10: $perplexity_{user-defined} += increase\_pre\_defined\_number$
11: re_compute p_{ij}
12: **end while**
13: **end if**
14: **end if**

The two strategies are applied in different situations to ensure the delivery of an output. Generally, the perplexity strategy works for all kinds of datasets. However, as the intrinsic grid search process for a big dataset is time-consuming, the Gaussian kernel strategy can be considered to deliver a result with any perplexity initialization. In our following experiments, the Gaussian kernel strategy is applied to large graphs, and perplexity increase strategy is applied to small graphs.

3.2 Appropriate perplexity estimation for the standard t-SNE

First of all, we would like to identify the relationship between a perplexity and the graph layout quality measured by the normalized stress, neighborhood preservation, and visualization that directly reflects the global connective structure.

The concept of "large" and "small" in terms of perplexity is dependent on the size of dataset; therefore, we map the value of perplexity to a specific percentage of the dataset size. The t-SNE perplexity defines the size of neighborhood within which the data points are considered. Based on the above discussion, we propose the following hypothesis:

Hypothesis: *A relative smaller perplexity will generate a graph layout with both higher normalized stress and more neighborhood preservation, while a larger perplexity will generate a graph layout with both lower normalized stress and less neighborhood preservation.*

As a higher normalized stress means more overall information will be lost from a global perspective, and a higher neighborhood preservation means neighborhood information can be preserved with a higher precision from a local perspective, and vice versa, a further question based on the hypothesis is, how to find the balance between a perplexity and graph layout quality measured by the two metrics. We propose our perplexity estimation approach by considering the size of dataset, the distribution of input data, and the graph density, which is illustrated as follows.

Given a graph $G(V, E)$, the Gaussian mean (μ) and kernel (σ) of the graph-theoretic distances of G is obtained at first. Then, we estimate a reasonable perplexity $perp$ according to the dataset size $|V|$, graph density $d = |E|/|V|$, μ and σ of the Gaussian, which can be described as:

$$\text{perp} = \begin{cases} a\_\text{fixed\_number}, & \text{if } |V| < \text{threshold}, \\ |V|\frac{\mu-2\sigma}{\mu}\delta(d), & \text{if } |V| \geq \text{threshold}, \end{cases} \tag{9}$$

where $\delta(d)$ can be regarded as the graph density regulator, as the graph density d is considered as a positively correlated factor of perplexity in addition to the dataset size, a small or large d corresponds to a small or large perplexity, respectively.

The idea behind the upper part of the perplexity selection in Eq. 9 is that the statistical result for small sized data is less reliable compared to relative large sized data. Therefore, we can set a fixed number as the perplexity. The idea behind the bottom part of the selection method can be explained from three aspects. First, the perplexity can be bigger for large dataset, smaller for small dataset, then the dataset size $|V|$ is a direct factor when choosing a perplexity. Second, the perplexity can be bigger for more densely distributed data, whereas a smaller perplexity works for sparsely distributed data. This factor can be measured by how far the Gaussian distribution spreads considering the neighborhood far away, i.e., $\frac{\mu-2\sigma}{\mu}$. Finally, the perplexity is regulated by the graph density regulator $\delta(d)$.

3.3 Appropriate perplexity estimation for sklearn Barnes–Hut TSNE

Moreover, we also consider the runtime issue. The runtime on large datasets such as graphs with over 2000 vertices are reported over several hundreds of seconds when applying tsNET and tsNET* (see Table 4 in [6]), similar trend about runtime for our modified version of the standard t-SNE is also expected. In the current Python environment, an accelerated t-SNE version, sklearn Barnes–Hut TSNE, is available by implementing Barnes–Hut approximations, allowing the tool to be applied on large real-world datasets [15].

We then adapt our perplexity estimation to sklearn Barnes–Hut TSNE by updating Eq. 9 as:

$$\text{perp} = |V|\frac{\mu-\sigma}{3\mu}\delta(d), \text{ if } |V| \geq \text{threshold}. \tag{10}$$

The adaption to sklearn Barnes–Hut TSNE in Eq. 10 is expected to work on the large datasets for accelerating purpose without losing too much precision. We test all the proposed methods in the following experiments.

4 Experiment and results

The visualizations presented in this paper are generated in Python with libraries including numpy, sklearn, networkx and plotly, in jupyter notebook or other web-based Python frameworks. We visualize the nodes in a graph as red dots, and the edges in red, green and grey with respect to different edge length. Equation 11 illustrates how the edge color is assigned, where e, color_e, d_μ and d_σ represent the length of the edge, color of the edge, mean distance and standard deviation of the graph-theoretic distances of the individual graph dataset, respectively.

$$\text{color}_e = \begin{cases} \text{red}, & \text{if } e < d_\mu - d_\sigma, \\ \text{green}, & \text{if } d_\mu - d_\sigma \leq e \leq d_\mu + d_\sigma, \\ \text{grey}, & \text{if } e > d_\mu + d_\sigma. \end{cases} \tag{11}$$

4.1 Test data

We test our approach with the 20 datasets released by the tsNET team [22]. Table 1 lists the datasets.

In order to analyze the results, we label the datasets regarding their size and graph density. Small and large graphs refer to graphs with less than 1000 and more than 1000 vertices, respectively. Sparse and dense graphs stand for graphs with graph density less than 3 and more than 3, respectively, where the graph density is calculated by $|E|/|V|$.

Table 1 Benchmark datasets

Dataset	Dataset type	$\|V\|$	$\|E\|$	$\|E\|/\|V\|$
dtw_72	Planar, structural	72	75	1.0417
lesmis	Collaboration network	77	254	3.2987
can_96	Mesh-like, structural	96	336	3.5000
rajat11	Miscellaneous	135	377	2.7926
jazz	Collaboration network	198	2742	13.8485
visbrazil	Collaboration network	222	236	1.5135
grid17	Planar, mesh-like, structural	289	544	1.8824
mesh3e1	Mesh-like, structural	289	800	2.7682
netscience	Collaboration network	379	914	2.4116
dwt_419	Structural	419	1572	3.7518
price_1000	Planar, tree	1000	999	0.9990
dwt_1005	Structural	1005	3808	3.7891
cage8	Miscellaneous	1015	4994	4.9202
bcsstk09	Mesh-like, structural	1083	8677	8.0120
block_2000	Clusters	2000	9912	4.9560
sierpinski3d	Structural	2050	6144	2.9971
CA-GrQc	Collaboration network	4158	13, 422	3.2280
EVA	Collaboration network	4475	4652	1.0396
3elt	Planar, mesh-like	4720	13, 722	2.9072
us_powergrid	Structural	4941	6594	1.3345

4.2 Validation of hypothesis

We carry out a series of tests to validate hypothesis. The perplexity is set as 2%, 5%, 10%, 20%, 30%, 40% and 50% of the size of individual dataset. Figure 1 shows the boxplot of M_{np} and M_σ over the 20 datasets, where the mean value of each subgroup labeled by the percentage is dotted.

Figure 1b shows that the average neighborhood preservation M_{np} increases to the peak when perplexity is less than 10% of dataset size, then decreases with the increase in perplexity. The average normalized stress M_σ in Fig. 1a increases to the peak when perplexity is less than 5% of dataset size, then decreases with the increase in perplexity. However, the decrease in M_σ is less sharp than the M_{np}, as shown in Table 2 where the linear model coefficient lm_coef of M_σ and M_{np} is -4.3826 and -6.1940, respectively. If we flip over one of the fitted lines against the x-axis, the intersection point of the two lines can be identified when $x = 0.4664$. It indicates that a large perplexity no more than 47% of the dataset size could result in quite *stable* visualization in which both normalized stress and neighborhood preservation could be more likely to be well-balanced. The correlation between pairs of variables is presented in Table 2. Pearson's correlation coefficient presented as cor_coef suggests that both of the average M_{np} and the average M_σ are significantly negatively correlated to perplexity, with correlation coefficient -0.8299 and -0.9450, respectively, both are very close to -1. The results strongly support the hypothesis that a smaller perplexity corresponds to a larger M_σ and a larger M_{np}, and that a larger perplexity corresponds to smaller M_σ and M_{np}. In addition, the smoothed quadratic means of both M_σ and M_{np} shown in Fig. 2, which are fitted in R by loess method and formula $y \sim x$, is also a strong visual support to the hypothesis.

To illustrate the issue visually with examples, Fig. 3 demonstrates two datasets with gradually increased perplexities ranging from 2 to 100% of the dataset size using the modified standard t-SNE. We can see that the visualizations generated with a larger perplexity are fairly robust from a global perspective (less M_σ), meanwhile preserving neighborhood information with less precision (less M_{np}) as more nodes are overlapped with the increase in perplexity.

We also examine the tsNET and tsNET* visualizations using their released code with the change of perplexities the same as in Fig. 3, which is shown as Fig. 4. The maximal iteration number is set 1100 for all tests presented in Fig. 3 and Fig. 4. A similar trend as Fig. 3 presented can be inspected from the graph layouts presented in Fig. 4 that tsNET and tsNET* visualizations generated with a larger perplexity are quite robust from the global perspective. However, the nodes of dw_1005 in tsNET and tsNET* layouts with a larger perplexity are much compressed with the increase in perplexity before 20% of the dataset size, and spread out with less overlapping with the increase in perplexity after 30% of the dataset size, which is different from what we can observe in

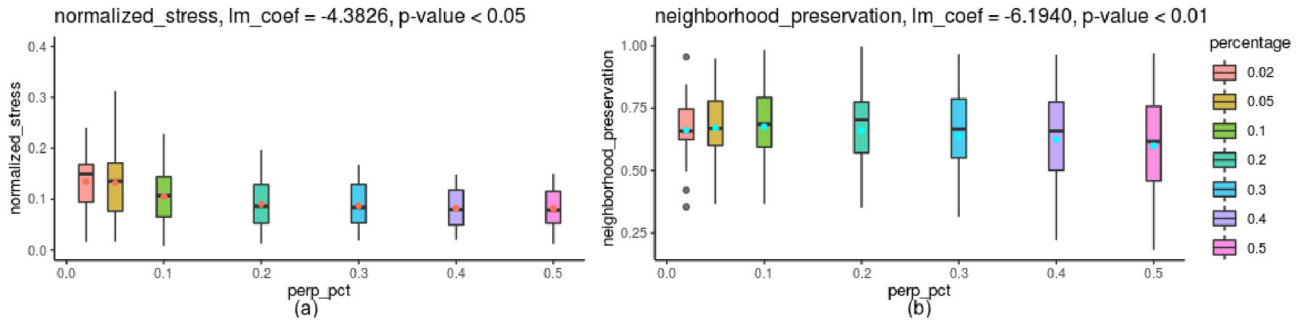

Fig. 1 Boxplot of neighborhood preservation and normalized stress of the 20 datasets, mean value of each percentage group is dotted

Table 2 Correlation of perplexity percentage and average M_σ and average M_{np}

Perp_pct with	lm coefficient and intercept		Correlation		
	lm_coef[1]	lm_intc[2]	cor_coef[3]	95% CI	p_value[4]
Average M_σ	− 4.3826	0.7024	− 0.8299	[− 0.9741, − 0.2048]	0.0209
Average M_{np}	− 6.1940	4.2302	− 0.9450	[− 0.9921, − 0.6655]	0.0013

[1] lm_coef is the fitted linear model coefficient
[2] lm_intc is the fitted linear model intercept
[3] cor_coef is the Pearson's correlation coefficient
[4] We adopt that values within 95% confidence interval (CI) and a p-value less than 0.05 are acceptable

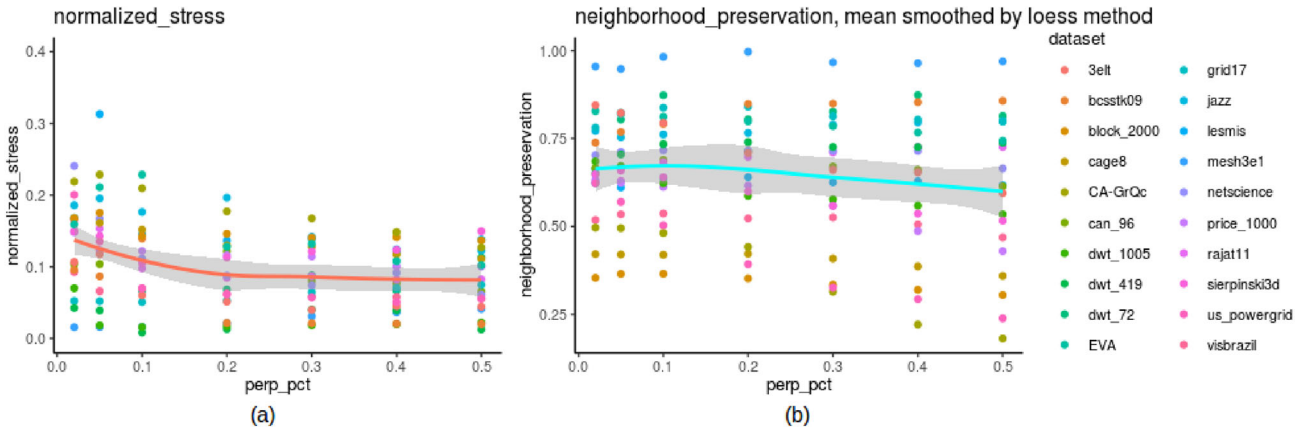

Fig. 2 Trends of neighborhood preservation and normalized stress of the 20 datasets with the increase in perplexity

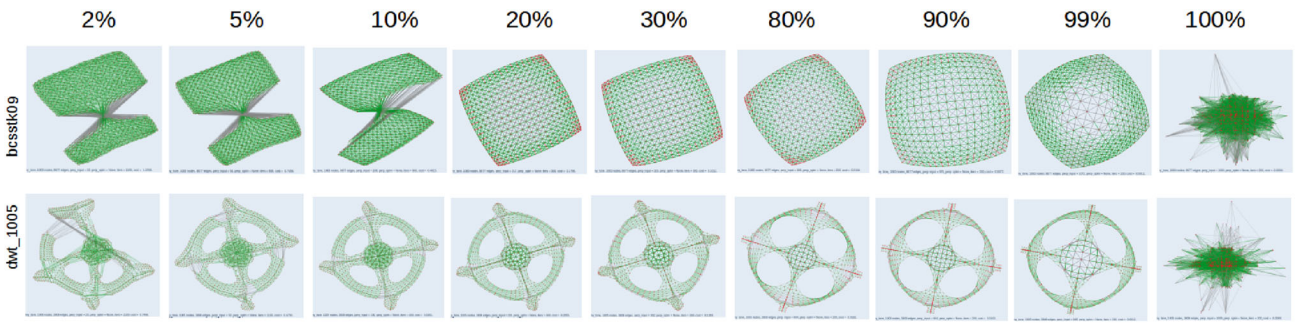

Fig. 3 Visualizations with perplexities as different percentages of the dataset size by the modified standard t-SNE

Fig. 3, most probably due to the additional control of node repulsion and edge attraction in tsNET [6].

Overall, we can identify the descending trends of both M_{np} and M_σ with the increase in perplexity in Fig. 1, with which the hypothesis can be validated. In addition, as shown in Table 2, M_{np} and M_σ are positively correlated with $p = 0.63$, it suggests that there is a need to find a trade-off between M_{np} and M_σ with respect to an appropriate perplexity, which will be large enough to generate a good layout with less M_σ, but small enough to preserve more neighborhood details with higher M_{np}.

4.3 Perplexity estimation based on our approach based on the modified standard t-SNE

We test our perplexity estimation approach with the 20 datasets. The perplexity is set according to Eq. 9, with the fixed value 40 as perplexity for small sized graphs. The threshold is 1000. Further, we roughly set the value of $\delta(d) \in \{0.1, 0.3\}$ in our experiment as described in Eq. 12, and only apply a big $\delta(d)$ value to large and very dense graphs to avoid overfitting.

$$\delta(d) = \begin{cases} 0.1, \text{ if } d < 6, \\ 0.3, \text{ if } d \geq 6 \text{ and } |V| \geq 1000. \end{cases} \tag{12}$$

4.3.1 M_σ and M_{np}

Table 3 shows the experiment results in details. Overall, we can find that both the average M_σ and M_{np} of our approach show improvements compared with the overall averages of tsNET* reported in [6]. It suggests that our perplexity estimation approach is very robust and effective to balance between normalized stress and neighborhood preservation. When observing the performance of our method on the four different types of graphs, the average M_σ and M_{np} for small- and large-sized groups as well as the sparse group also demonstrate excellent performance stability. The performance of our approach on dense graphs is also comparable to that of tsNET*, with a less average normalized stress and neighborhood preservation, without significant difference however. The performance on several individual datasets including dwt_72, rajat11, 3elt, us_powergrid and dwt_1005 outperform that of tsNET*, the others show either a better or comparable M_σ or a better or comparable M_{np} at the same time, except dwt_419. Then, we test with a larger perplexity, a value which is over 15% of the dataset size of dwt_419, we can obtain very stable excellent performance.

4.3.2 Visual inspection

Visual inspections of all datasets with perplexity of 2% and 10% of dataset size, as well as estimation with our method based on the modified standard t-SNE, are shown in Fig. 6. We can find that a small perplexity often results in a decrease/uncertainty of fitting precision and therefore causes visual distortions of the graph layout. Several mesh-type graph examples in Fig. 6, such as dwt_72, can_96, 3elt, dwt_1005, sierpinski3d and bcsstk09 can illustrate this problem clearly, when their perplexities are set as small as 2% of the individual dataset size. Overall, most of the visualizations generated with the perplexity estimated with our approach based on the modified standard t-SNE, except dwt_419, display better structures without or with less visual distortions (shown as gray edges). dwt_419 can be better visualized when the perplexity is increased, as mentioned above.

4.3.3 Runtime

The runtime in seconds is shown in Fig. 5, where the datasets are ordered by their number of nodes (size) along the x-axis. We do not compare the runtime in a precise way the same as in Table 3 due to the reason that the runtime reported in the work of tsNET and tsNET* was based on their customized settings of perplexity and stop conditions, which are unknown to us. We can observe that the runtime in Fig. 5 in general is increased with the decrease in perplexity from 2 to 50% for the same dataset, and large datasets need more time. It shows that our perplexity estimation approach chooses the perplexity between 2 and 10% of dataset size for most of the datasets tested.

We focus on the performance of our approach based on the modified standard t-SNE rather than the speed in the experiments above. As it takes several hundreds of seconds for tsNET/tsNET* and our modified version of the standard t-SNE on large datasets with over 4000 nodes, we try to improve it by the adapted estimation method as described in Eq. 10. Our experiment on the large test datasets also demonstrates very good performance with $M_\sigma = 0.1218$ and $M_{np} = 0.6296$ on average, compared to the average tsNET* $M_\sigma = 0.1243$ and $M_{np} = 0.5971$, as given in Table 4. We find that our perplexity estimation for the Barnes–Hut t-SNE does not work well on small datasets, and it works better on large dense data than on large sparse data, as the Barnes–Hut approximation on sparse or small data causes a higher information loss. The average runtime is dramatically reduced from 598 s with our modified standard t-SNE to 12.9 s with sklearn Barnes–Hut TSNE, while keeping overall good graph layout quality comparable to the method based on the modified standard t-SNE.

4.4 Discussion

In our approach, we estimate the value of perplexity based on the normality of the input data, which is very effective when tested on the benchmark datasets. We also notice that

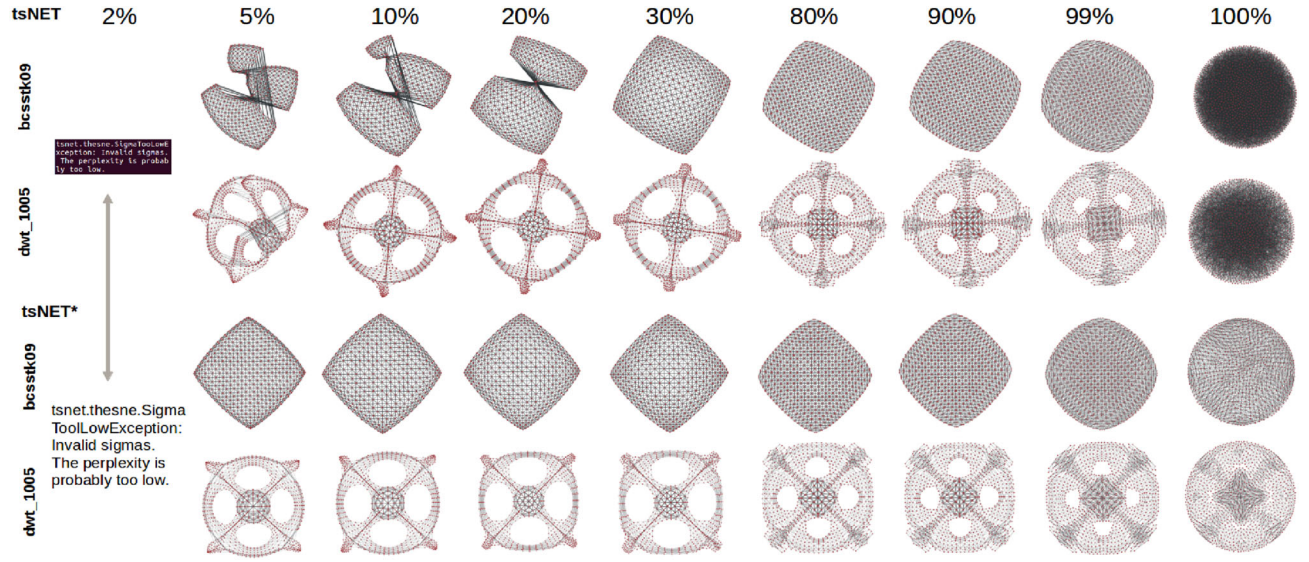

Fig. 4 tsNET and tsNET* visualizations with perplexities as different percentages of the dataset size

Table 3 Normalized stress (M_σ) and neighborhood preservation (M_{np}) measures of graph layouts by the modified standard t-SNE (st-SNE), where italics and bold numbers correspond to outperformed and underperformed measures compared to tsNET*, respectively

Dataset	Size/density label	M_σ^1		M_{np}^2	
		tsNET*	st-SNE	tsNET*	st-SNE
Perplexity $= 40$					
dtw_72	Small/sparse	0.048	**0.0392**	0.855	**0.8647**
rajat11	Small/sparse	0.097	**0.0779**	0.717	**0.7220**
visbrazil	Small/sparse	0.098	**0.0722**	0.584	*0.5308*
grid17	Small/sparse	0.021	*0.0493*	0.785	**0.8499**
mesh3e1	Small/sparse	0.014	*0.0166*	0.904	**0.9984**
netscience	Small/sparse	0.100	*0.1112*	0.707	**0.7154**
lesmis	Small/dense	0.111	**0.0967**	0.712	*0.6945*
can_96	Small/dense	0.084	**0.0743**	0.671	*0.6495*
jazz	Small/dense	0.128	*0.1388*	0.804	**0.8077**
dwt_419	Small/dense	0.024	*0.0386*	0.741	*0.7108*
Perplexity $= \lvert V\rvert \frac{\mu-2\sigma}{\mu}\delta(\frac{\lvert E\rvert}{\lvert V\rvert})$ as in Eq. 9					
price_1000	Large/sparse	0.160	**0.1220**	0.639	*0.6229*
sierpinski3d	Large/sparse	0.093	*0.1013*	0.580	**0.6531**
EVA	Large/sparse	0.161	*0.2161*	0.802	**0.8227**
3elt	Large/sparse	0.090	**0.0717**	0.715	**0.8113**
us_powergrid	Large/sparse	0.101	**0.0742**	0.457	**0.5424**
dwt_1005	Large/dense	0.035	**0.0185**	0.619	**0.6667**
cage8	Large/dense	0.203	**0.1658**	0.437	*0.4219*
bcsstk09	Large/dense	0.022	0.0220	0.867	*0.8509*
block_2000	Large/dense	0.189	**0.1604**	0.372	*0.3640*
CA-GrQc	Large/dense	0.189	*0.2057*	0.483	**0.4871**
Average	Overall	0.0984	**0.0936**	0.6726	**0.6893**
	Small	0.0725	**0.0715**	0.7480	**0.7544**
	Large	0.1243	**0.1158**	0.5971	**0.6243**
	Sparse	0.0894	**0.0865**	0.7041	**0.7394**
	Dense	0.1094	**0.1023**	0.6340	*0.6281*

[1]The smaller, the better M_σ is, whereas the bigger, the better M_{np} is
[2]The M_σ and M_{np} of tsNET* are drawn from [6]
[3]Values in bold/italics stand for measures outperform/underperform tsNET*

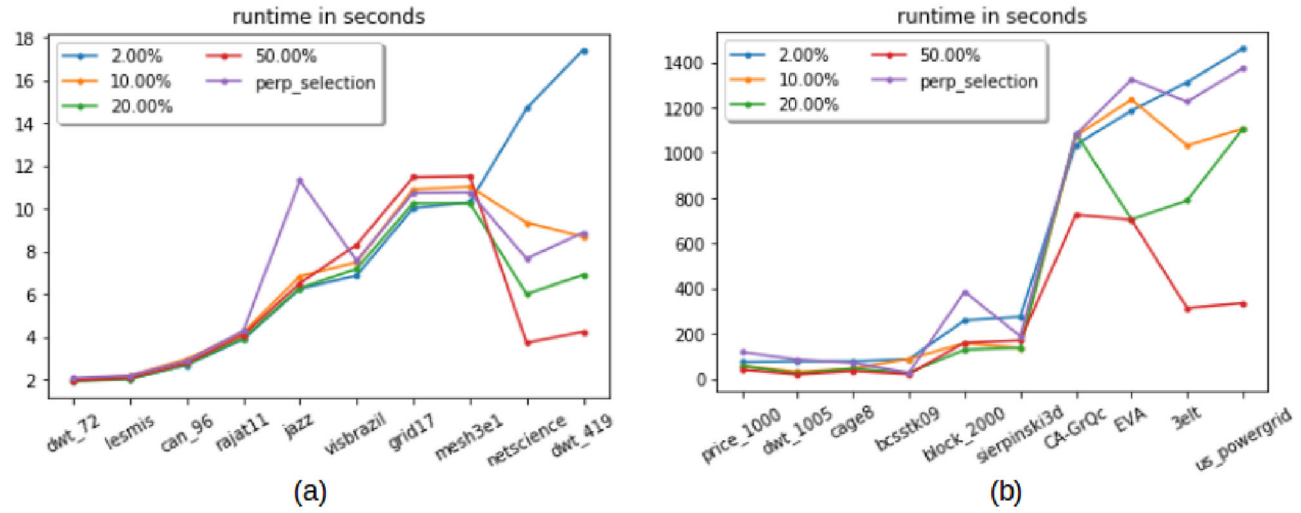

Fig. 5 Runtime in seconds with different perplexity as percentage of dataset size based on the modified standard t-SNE (**a** small datasets, **b** large datasets)

Table 4 Normalized stress (M_σ), neighborhood preservation (M_{np}) measures and runtime (in seconds) of sklearn Barnes–Hut TSNE (TSNE)

Perplexity $= \lvert V\rvert \frac{\mu-\sigma}{3\mu}\delta(\frac{\lvert E\rvert}{\lvert V\rvert})$ as in Eq. 10						
	Dataset	TSNE runtime (s)	M_σ		M_{np}	
			tsNET*	TSNE	tsNET*	TSNE
Large sparse	price_1000	2.8	0.160	*0.1666*	0.639	*0.6332*
	sierpinski3d	5.0	0.093	*0.1421*	0.580	**0.6538**
	EVA	27.4	0.161	*0.1771*	0.802	**0.8197**
	3elt	23.5	0.090	**0.0660**	0.715	**0.8226**
	us_powergrid	24.8	0.101	**0.0889**	0.457	**0.5789**
Large dense	dwt_1005	2.4	0.035	**0.0199**	0.619	**0.6744**
	cage8	2.5	0.203	**0.1551**	0.437	*0.4161*
	bcsstk09	3.5	0.022	*0.0242*	0.867	*0.8420*
	block_2000	5.6	0.189	**0.1519**	0.372	*0.3645*
	CA-GrQc	31.3	0.189	*0.2264*	0.483	**0.4915**
Average	Large-overall	12.9	0.1243	**0.1218**	0.5971	**0.6296**
	Large-sparse	16.7	0.1210	*0.1281*	0.6386	**0.7014**
	Large-dense	9.0	0.1276	**0.1155**	0.5556	**0.5577**

Values in bold/italics stand for measures outperform/underperform tsNET*

each test dataset is a connected component such that the distribution of pairwise graph-theoretic distances fits normality well. For graphs with many disconnected components, a layout with much higher stress is supposed to be generated by t-SNE, as pilot experiments show, and similar findings is also reported in the work [7]. Focusing on the largest connected component or the connected component of interest is a practical solution to employ t-SNE on graphs with many disconnected components.

We estimate the value of perplexity for small sized graph with less than 1000 nodes to 40, which is an approximation between 2 and 10% of 1000, based on the average graph size of the test data. An increased value of graph density regulator $\delta(d)$ is not applied to the small graphs to avoid overfitting, whose visualizations show more randomness/uncertainty than the large graphs due to the data sparsity. In our experiment, we observe an automatic increase in perplexity from 40 to 100 for the small dense dataset *jazz*, and it shows the robustness of our approach.

Our approach presents advantages with respect to ease of use and effectiveness.

First, our approach does not require users to try multiple times with different perplexities as input to deliver the acceptable output, which is very easy to use. For example, the EVA dataset can be visualized with any smaller perplexities compared to 600 in tsNET and tsNET*, as shown in

Fig. 6. However, an extra grid search could probably help to find an optimum with extra costs of time and effort.

Second, our approach does not rely on the t-SNE parameter tuning such as the weight of the compression term [6,7], and it does not require an additional PivotMDS layout as initialization as tsNET* does, although the employment of a PivotMDS layout seems to be beneficial for small graphs as tested. As the tsNET* works based on the initialization of PivotMDS layouts, t-SNE cannot obtain better embeddings in case the PivotMDS layouts could not fully reflect the relations based on graph-theoretic distances.

Experimental results show that our approach is very effective to visualize graph data with good quality evaluated by normalized stress, neighborhood preservation and visual inspections, as well as a significant improvement in runtime when applied with sklearn Barnes–Hut TSNE on large datasets without losing visualization quality. Our work is a step beyond the work of Kruiger et al. [6] and Wattenberg et al. [13], and it not only reveals the impact of t-SNE perplexity on graph layouts, but also presents guidelines to estimate an appropriate perplexity for good layout, and offers possibility to accelerate the process using widely used Python tools.

5 Conclusion and future work

In this paper, we investigated the relationship of t-SNE perplexity and graph layout, improved the standard t-SNE to fit a variety of perplexity initialization, and proposed a perplexity estimation approach for good-quality graph data visualization evaluated by widely recognized quality metrics of graph layout.

Our approach demonstrated effectiveness and ease of use for graph data visualization when tested on a set of different types of benchmark datasets. As t-SNE is a widely recognized DR technique and perplexity is the most significant parameter, our research can be beneficial to a broad range of related researches and applications.

In the future, we will investigate other graph structures to improve our estimation, especially the density regulator, and test more types of data such as the disconnected networks and labeled networks. We are also very interested in investigating the relationship of similar parameters as perplexity and graph layout in other manifold algorithms such as UMAP.

Statement

Note: "I" refers to the corresponding author Chun Xiao in the following statement.

- All authors wrote and reviewed the main manuscript, Chun Xiao prepared the figures.
- I confirm that I understand journal International Journal of Data Science and Analytics is a transformative journal. When research is accepted for publication, there is a choice to publish using either immediate gold open access or the traditional publishing route.
- I declare that the authors have no competing interests as defined by Springer, or other interests that might be perceived to influence the results and/or discussion reported in this paper.
- The results/data/figures in this manuscript have not been published elsewhere, nor are they under consideration by another publisher.
- I have read the Springer journal policies on author responsibilities and submit this manuscript in accordance with those policies.
- All of the material is owned by the authors and/or no permissions are required.
- This research did not involve Human Participants and/or Animals.

Acknowledgements This research was supported by the Australian Research Council Linkage Grant LP160100935 and Oracle Research Lab Australia.

Funding Open Access funding enabled and organized by CAUL and its Member Institutions

Declarations

Conflicts of interest The authors declare no competing interests.

Appendix A Visual inspections of test datasets

See Fig. 6

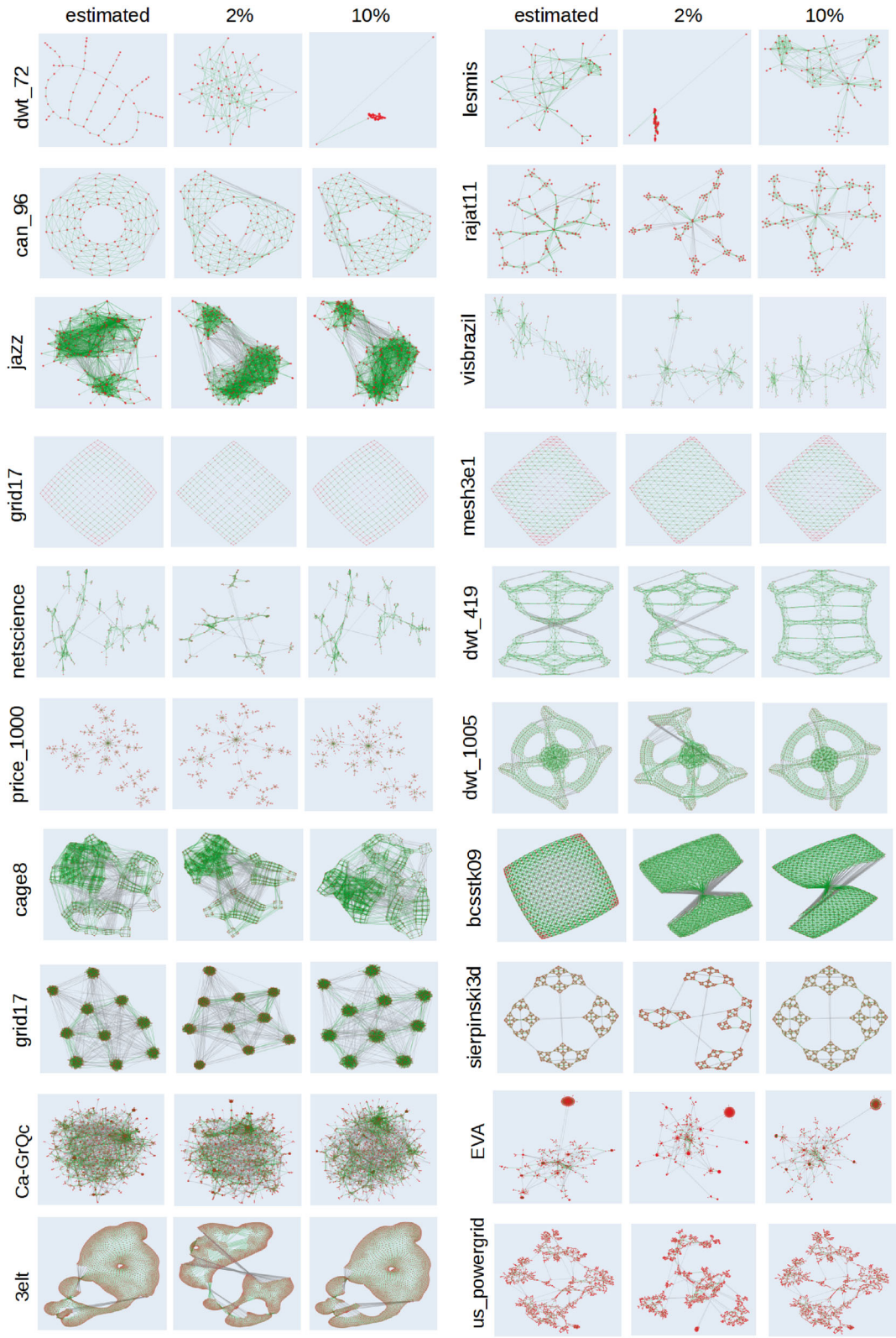

Fig. 6 Visual inspections of the 20 datasets tested. Individual dataset is visualized with estimated perplexity by our approach on the modified standard t-SNE (estimated), and 2% and 10% of dataset size as perplexity

References

1. Buriánek, T., Zaorálek, L., Snášel, V., Peterek, T.: Graph drawing using dimension reduction methods. Adv. Intell. Syst. Comput. **334**, 205–214 (2015). https://doi.org/10.1007/978-3-319-13572-4_17
2. Zaorálek, L., Buriánek, T., Snášel, V.: Dimension reduction methods in graph drawing problem. In: 2014 14th International Conference on Intelligent Systems Designand Applications, pp. 13–18. IEEE, USA (2014).https://doi.org/10.1109/ISDA.2014.7066252
3. Harel, D., Koren, Y.: Graph drawing by high-dimensional embedding. In: Revised Papers from the 10th International Symposium on Graph Drawing. GD '02, pp. 207–219. Springer,Berlin, Heidelberg (2002).https://doi.org/10.5555/647554.757122
4. Brandes, U., Pich, C.: Eigensolver methods for progressive multidimensional scaling of large data. In: Proceedings of the 14th International Conference on Graph Drawing. GD'06, pp. 42–53. Springer, Berlin, Heidelberg (2006). https://doi.org/10.5555/1758612.1758620
5. Zhu, M., Chen, W., Hu, Y., Hou, Y., Liu, L., Zhang, K.: Drgraph: an efficient graph layout algorithm for large-scale graphs by dimensionality reduction. IEEE Trans. Vis. Comput. Gr. **27**(2), 1666–1676 (2021). https://doi.org/10.1109/TVCG.2020.3030447
6. Kruiger, J.F., Rauber, P.E., Martins, R.M., Kerren, A., Kobourov, S., Telea, A.C.: Graph layouts by t-SNE. Comput. Graph. Forum **36**(3), 283–294 (2017). https://doi.org/10.1111/cgf.13187
7. Xu, G., Song, Z., Wang, Y., Lin, D., Chen, J., Mao, T., Xu, W.: A graph layout framework combining t-distributed neighbor retrieval visualizer and energy models. IEEE Access **7**, 27515–27525 (2019). https://doi.org/10.1109/ACCESS.2019.2900358
8. Leow, Y.Y., Laurent, T., Bresson, X.: Graphtsne: A visualization technique for graph-structured data. In: ICLR Workshop on Representation Learning on Graphs and Manifolds, pp. 1–7. https://iclr.cc/, online (2019)
9. Kamada, T., Kawai, S.: An algorithm for drawing general undirected graphs. Inf. Process. Lett. **31**(1), 7–15 (1989). https://doi.org/10.1016/0020-0190(89)90102-6
10. Yang, Z., Peltonen, J., Kaski, S.: Optimization equivalence of divergences improves neighbor embedding. In: Proceedings of the 31 St International Conference on Machine Learning, vol. 32, pp. 460–468. JMLR.org, USA (2014). https://doi.org/10.13140/2.1.3716.9281
11. van der Maaten, L., Hinton, G.: Visualizing data using t-SNE. J. Mach. Learn. Res. **9**, 2579–2605 (2008)
12. Espadoto, M., Martins, R.M., Kerren, A., Hirata, N.S.T., Telea, A.C.: Towards a quantitative survey of dimension reduction techniques. IEEE Trans. Vis. Comput. Gr. Press (in Press) **27**(3), 2153–2173 (2019). https://doi.org/10.1109/TVCG.2019.2944182
13. Wattenberg, M., Viégas, F., Johnson, I.: How to use t-SNE effectively. Distill Online (2016). https://doi.org/10.23915/distill.00002
14. Github t-SNE. https://lvdmaaten.github.io/tsne
15. van der Maaten, L.: Accelerating t-SNE using tree-based algorithms. J. Mach. Learn. Res. **15**, 1–21 (2014)
16. Venna, J., Peltonen, J., Nybo, K., Aidos, H., Kaski, S.: Information retrieval perspective to nonlinear dimensionality reduction for data visualization. J. Mach. Learn. Res. **11**, 451–490 (2010). https://doi.org/10.5555/1756006.1756019
17. de Rosa, G., Brega, J.R., Papa, J.: How optimizing perplexity can affect the dimensionality reduction on word embeddings visualization? SN Appl. Sci. (2019). https://doi.org/10.1007/s42452-019-1689-4
18. de Bodt, C.,Mulders, D., Verleysen, M., Lee, J.: Perplexity-free t-SNE and twice student tt-SNE. In: Proceedings of European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2018, pp. 123–128. i6doc, Louvain-la-Neuve, Belgium (2018)
19. Crecchi, F., de Bodt, C., Verleysen, M., Lee, J., Bacciu, D.: Perplexity-free parametric t-SNE. In: Proceedings of European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2020, pp. 387–392. i6doc, Louvain-la-Neuve, Belgium (2020)
20. de Bodt, C., Mulders, D., López-Sánchez, D., Verleysen, M., Lee, J.: Class-aware t-SNE: cat-SNE. In: Proceedings of European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2019, pp. 409–414. i6doc, Louvain-la-Neuve, Belgium (2019)
21. Chatzimparmpas, A., Martins, R., Kerren, A.: t-visne: interactive assessment and interpretation of t-SNE projections. IEEE Trans. Vis. Comput. Gr. **26**(8), 2696–2714 (2020)
22. Github tsNET. https://github.com/HanKruiger/tsNET/