Multivariate Graph Layouts using NNP-NET

I. Hartskeerl[1] and F. Grötschla[2] and A. C. Telea[1]

[1]Department of Information and Computing Science, Utrecht University, Utrecht, Netherlands
[2]ETH Zurich, Zürich, Switzerland

Abstract

Laying out graphs with node attributes so as to reflect both the graph topology and node attribute similarity is an ongoing challenge in graph drawing. We approach this task by extending NNP-NET, a recent graph drawing method based on deep-learning dimensionality reduction, to incorporate the similarity of attributes into its cost function. Our simple modification preserves all desirable properties of NNP-NET such as ability to handle any graphs, out-of-sample extension, and linear cost in the input size. We offer a single parameter to users to smoothly navigate between layouts that show only the graph data, attribute data or a combination of both. We illustrate our method on several real-world graphs annotated with high-dimensional attributes.

CCS Concepts

• ***Mathematics of computing*** → ***Dimensionality reduction;*** • ***Computer systems organization*** → ***Neural networks;*** • ***Human-centered computing*** → *Graph drawings;*

1. Introduction

Many techniques exist for drawing graphs such as force-directed [FR91], spectral [BP07], and deep learning [GMVW24, GLA*21, TCG22]. Drawing graphs is closely related to dimensionality reduction (DR) which projects high-dimensional data into a low-dimensional space (typically 2D or 3D). This is shown by DR methods such as MDS [Tor52] and t-SNE [vdMH08] that can be readily adapted to draw graphs (for the latter, see tsNET [KRM*17]).

Multivariate graphs have nodes with one or more attributes. When drawing such graphs, one aims to position nodes close to each other if these either have similar attributes or are (strongly) connected in the graph [VBW17]. A few classical techniques have been extended to use attributes, *e.g.*, force-directed [GFV13], DR-based [GV17], and deep learning [STSW24] approaches.

Recently, deep learning has been used to accelerate DR by the NNP (Neural Network Projections) method [EHST20] which is supervised by a projection of a subset of the data done by a user-selected DR technique such as t-SNE. NNP-NET followed similarly to accelerate tsNET [HMvW*25]. NNP-NET has several attractive properties – it produces high-quality layouts for graphs of any topology, handles edge weights, has the out-of-sample property, is stable to small input changes, and has a time cost linear in the input graph size. Yet, it cannot handle multivariate graphs.

We present a modification of NNP-NET to handle multivariate graphs. Our method preserves all the mentioned desirable properties of NNP-NET, can handle any number of node attributes, allows a smooth user-controlled transition between a pure topology (graph-drawing-like) and a pure attribute-based (DR-like) drawing, all with limited computational cost and keeping linear time complexity.

2. Method

Let $G = (V, E)$ be a graph with nodes $V = \{v_i\}_{i=1}^N$ and edges $E = \{(v_i, v_j) \in V \times V\}$. With multivariate graphs, G also includes $X = \{\mathbf{x}_i\}_{i=1}^N$, where $\mathbf{x}_i \in \mathbb{R}^n$ is a n-dimensional feature vector or attribute.

As we extend NNP-NET [HMvW*25], let us next outline its five key steps: (1) A DR method (specifically, PMDS [BP07] which is simple and fast) is used to map G's graph-theoretic distance matrix to $d = 50$ dimensions. (2) A subgraph G' of (roughly) 10K nodes is extracted from G. (3) tsNET is used to lay out G' in 2D creating a so-called ground truth. (4) NNP (Neural Network Projections) [EHST20] is trained to learn the mapping from $\mathbb{R}^d$ (coordinates created by PMDS) to $\mathbb{R}^2$ (ground truth layout created by tsNET). (5) After training, NNP is run on the full d-dimensional PMDS projection of G to lay out the entire G.

To add multivariate data X to this pipeline, steps 3-5 must be adapted. Step 1 only maps the graph *topology* to $\mathbb{R}^d$ so it does not depend on attributes. For step 2, NNP on high dimensional data would typically use *random* sampling, so accounting for attribute datais not needed. tsNET (step 3) internally uses t-SNE which takes an input matrix $\{G'_{ij}\}_{i,j=1}^N$, where G'_{ij} is the graph-theoretic distance between nodes v_i and v_j of G', to create a 2D embedding of G'. To account for attributes in this phase, we replace G'_{ij} with

$$\delta_{ij} = \alpha \|\mathbf{x}_i - \mathbf{x}_j\|^2 / \max_{i,j} \|\mathbf{x}_i - \mathbf{x}_j\|^2 + (1-\alpha) G'_{ij} / \max_{i,j} G'_{ij}, \quad (1)$$

where $\alpha \in [0, 1]$ is a user-set parameter that controls how much to

account for attributes in the layout. Attribute distances $\|\mathbf{x}_i - \mathbf{x}_j\|$ are computed using the squared L_2 norm. Equation 1 normalizes both attribute and graph-theoretic between $[0,1]$ to allow for a simple linear interpolation between them based on α.

Step 4 and 5 only need to be changed slightly. For each vertex $v_i \in G$, we take the d dimensions created by PMDS (see step 1), append to these the n-dimensional attribute $\mathbf{x}_i$, and use the resulting $d+n$ dimensional vector as input to train and infer NNP. Both this modification and the modification to step 3 do not change the time complexity of NNP-NET, which stays linear.

3. Results

We tested our method on five graphs of various sizes, provenances, and with attributes having different dimensionalities (Tab. 1). Class information is used only to color the resulting drawings for further exploration (see next Tab. 2). For the CIFAR-100, MNIST and Fashion-MNIST datasets, we first encoded each image into a $n = 512$-dimensional embedding using the pretrained CLIP ViT-B/32 model [RKH*21]. We then created a k-nearest neighbor graph (k=15) over the resulting embeddings using cosine distance, linking each node to its 15 closest neighbors in CLIP space. The remaining Cora, Pubmed and Reddit datasets are graphs that come with attributes from their respective sources. Class labels are not used during graph construction.

Dataset	$\|V\|$	$\|E\|$	$\|\mathbf{x}_i\|$	classes
Cora [YCS16]	2708	10556	1433	7
Pubmed [YCS16]	19717	44338	500	3
MNIST [Den12]	60000	1330650	512	10
Fashion-MNIST [XRV17]	60000	1355872	512	10
CIFAR-100 [Kri09]	50000	1138780	512	100
Reddit [HYL17]	232965	11606919	602	41

Table 1: *Datasets used for evaluation.*

We created 2D layouts for all datasets using $\alpha = \{0, 0.25, 0.5, 0.75, 1\}$. All other parameters follow NNP-NET defaults (see [HMvW*25], Tab. 2), except for NNP-NET's final smoothing step of which we turned off to get the raw layout results. Table 2 shows the resulting graph layouts with nodes colored by label. We see quite well formed clusters of same-color nodes which is expected for a good layout as, for these datasets, we know that connected nodes have similar labels. We next explore these results.

Cora: We see a clear difference between the layouts for $\alpha = 0$ and $\alpha = 1$. For $\alpha = 1$, edges become longer (see the larger amount of black in the figure); the drawing has less structure compared to using $\alpha = 0$. This is expected, as only attributes, and not the graph structure, is taken into account by $\alpha = 1$. Interestingly, using $\alpha = 0$ separates the classes better from each other than $\alpha = 1$ – see for instance the green class. This shows that the relational information (edges) in this graph is better correlated with class information than the attribute information. As α increases, the fine-grained structure of the layout given by the edges information is lost and nodes are spread in more uniform same-color structures.

Pubmed, Reddit: As with Cora, we see a large difference between the results for $\alpha = 0$ and $\alpha = 1$ results. Similarly, we see that

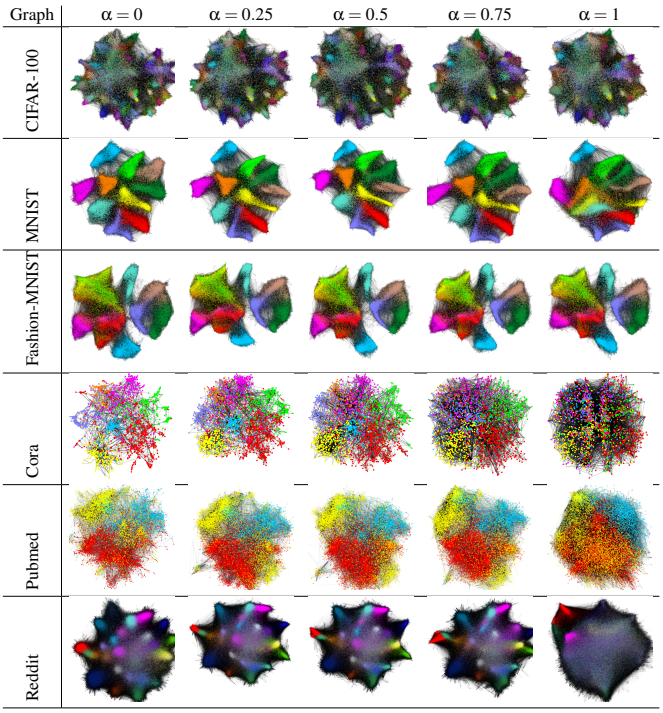

Table 2: *Layouts created by our method for different values for* α.

Dataset	Stress	$\alpha = 0$	$\alpha = 0.25$	$\alpha = 0.5$	$\alpha = 0.75$	$\alpha = 1$
CIFAR-100	σ_G	0.145	0.146	0.145	0.147	0.143
	$\sigma_{\mathbf{x}}$	0.179	0.176	0.173	0.178	0.166
MNIST	σ_G	0.144	0.146	0.147	0.144	0.144
	$\sigma_{\mathbf{x}}$	0.199	0.201	0.202	0.196	0.180
Fashion-MNIST	σ_G	0.116	0.113	0.119	0.119	0.115
	$\sigma_{\mathbf{x}}$	0.144	0.146	0.147	0.147	0.141
Cora	σ_G	0.163	0.174	0.181	0.222	0.310
	$\sigma_{\mathbf{x}}$	0.250	0.230	0.214	0.208	0.206
Pubmed	σ_G	0.188	0.196	0.197	0.199	0.223
	$\sigma_{\mathbf{x}}$	0.224	0.220	0.216	0.206	0.195
Reddit	σ_G	0.190	0.194	0.194	0.194	0.214
	$\sigma_{\mathbf{x}}$	0.430	0.431	0.428	0.417	0.290

Table 3: *Stress values (graph* σ_G *and attribute based* $\sigma_{\mathbf{x}}$*) from running our method with different values for* α*. Lower is better.*

$\alpha = 0$ achieves more visual separation of classes, again an indication that labels better correlate with the edges information than with attributes. All these findings are interesting at a higher level: Following [RFT17], who showed that class separation in a 2D DR plot is indicative of the ease of class prediction by the features used for projection, it would be likely challenging to train a classifier to predict labels purely based on attributes in these datasets (that is, without accounting for the edges).

MNIST, Fashion-MNIST, CIFAR-100: The effect of α is far less pronounced in these examples. This is expected since, as explained, we synthesized the graph information for these purely attribute datasets based on the attribute values – and more specifically, we linked nodes which correspond to k-nearest samples in attribute space. As such, the drawings highly resemble what a pure projection

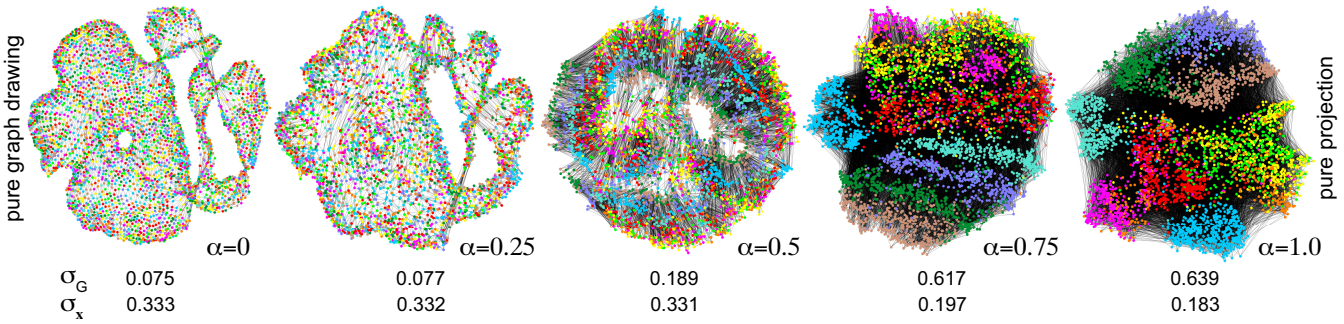

Table 4: 3elt *graph combined with features from MNIST, different* α *values. Each layout shows the graph and attribute stress* σ_G *and* $\sigma_{\mathbf{x}}$.

(DR plot) of the respective attribute datasets would be. This is most visible for MNIST – we recognize the well-known 'visual signature' of a t-SNE projection of this dataset in these drawings. Yet, edges *do* have an effect on the layout: For instance, for MNIST ($\alpha = 1$), the yellow cluster is split into two separate clusters separated by the orange cluster; for all other α values, this structure is one cluster.

We further analyze the effect of α by computing the *normalized stress* of our layouts. Specifically, to measure how well the drawing captures the graph structure encoded by the distance matrix G_{ij}, we compute the *graph stress* $\sigma_G = \min_s \frac{1}{|V|^2} \sum_{i \neq j} ((G_{ij} - s\Delta_{ij})/G_{ij})^2$, where Δ_{ij} is the 2D distance between nodes i and j in the drawing of G and s is a scaling factor to minimize the stress for a fair comparison. Furthermore, we compute the *attribute stress* $\sigma_{\mathbf{x}} = \min_s \frac{1}{|V|^2} \sum_{i \neq j} (((\|\mathbf{x}_i - \mathbf{x}_j\|^2 - s\Delta_{ij})/\|\mathbf{x}_i - \mathbf{x}_j\|^2)^2$, which measures how well the drawing captures the attributes X. Table 3 shows the σ_G and $\sigma_{\mathbf{x}}$ values for our experiments for different α values. We see that datasets which show larger visual changes as α varies (Tab. 2, *e.g.*, Cora and Pubmed) also have larger stress variations with α. Separately, we see how σ_G decreases as α decreases (as the graph information is favored) while $\sigma_{\mathbf{x}}$ decreases as α increases (as the attribute information is favored).

We further validate whether α value affects the layout in the expected way by an extra test. We take the *3elt* graph ($|V| = 4720$, $|E| = 13722$, SuiteSparse benchmark [DH11]) and assign to each of its nodes a *random* sample from the MNIST dataset. In contrast to the examples in Tab. 2, edges and attributes are now completely *uncorrelated*, so creating a layout which reflects both is significantly harder. Table 4 shows the resulting layouts for different α values. For $\alpha = 0$, our method recovers accurately the *expected* drawing of this graph – compare this image with those for the same graph in *e.g.* [KRM*17, HMvW*25]. Colors show here the random assignment of attributes to nodes. Hence, attributes have no effect in this case, as expected. For $\alpha = 0.75$, we see how same-color node clusters start forming as attributes become more important. For $\alpha = 1$, our method recovers accurately the *expected* t-SNE plot of MNIST – compare this image with the many such plots in the literature or Table 2. Additionally, we see that σ_G is low and $\sigma_{\mathbf{x}}$ is high for $\alpha = 0$, and conversely for $\alpha = 1$ – again, the result we desire from our method.

3.1. Computation time

Using attributes when drawing a graph results in extra cost compared to the 'raw' NNP-NET method which uses only the graph information. To measure this, we ran our method, implemented in C++ (code available online [HGT26]) on the MNIST dataset with, and without using, the attributes. The CPU used is a Intel Core Ultra 7 165H. Table 5 (a) shows the time for computing the ground-truth (GT) layout of subgraph G' (step 3, Sec. 2), NNP training + inference to lay out the entire graph G (step 4 and 5, Sec. 2), and the total time. For these tests, we used the exact t-SNE implementation from [vdMH08]. Including attribute data increases runtime by about 3%. For larger graphs, this difference will be even smaller, as the GT time is constant and the embedding creation time (step 1) which does not depend on X starts to dominate computation time.

Table 5 (b) shows timings of the same test when using tSNE's Barnes-Hut approximation (BH-tSNE, [vdM14]). As shown earlier in [HMvW*25], when not using attributes, using BH-tSNE is significantly faster than the exact version. When using attributes, the extra time cost is about 15%. This is because our current implementation needs to compute the full similarity matrix for G'.

Separately, we note the increase in NNP training and inference time in Tab. 5 when using attributes regardless of using exact- or BH-tSNE. This is directly explained by the fact that, when using attributes, NNP must input a (significantly) higher-dimensional vector – that is, having $d + n$ dimensions instead of d ($d = 50$ is the PMDS embedding space; n is the feature vector size, hundreds for MNIST, see Sec. 2).

	(a) exact t-SNE			(b) BH t-SNE		
Attributes	GT	NNP	Total	GT	NNP	Total
No	217.8	21.8	243.8	20.9	14.1	39.1
Yes	221.5	25.7	251.3	24.4	16.9	45.3

Table 5: *Execution time (seconds) of our method with and without using attributes, using exact and approximate (Barnes-Hut) t-SNE to compute the ground-truth (GT) layout, MNIST dataset.*

4. Discussion and Conclusion

Our method keeps the desirable properties of NNP-NET – being able to handle any graphs, scalable, out-of-sample extension (by construction, given the use of a trained NNP model), and the same parameter settings. The method for creating the ground-truth layout

(currently t-SNE) can be directly swapped with any other layout method that can handle multivariate graphs, leading to different drawing styles. While not directly tested, our method should also inherit NNP-NET's ability to handle edge weights; and the stability upon small input changes inherent to NNP-NET given the underlying stability of the NNP deep-learned model [BTT22].

Adding attributes needs only small changes to NNP-NET and increases total running cost by about 15% for all graphs tested. Our current BH-tSNE-based implementation has higher overhead as it needs the full similarity matrix of G', increasing ground-truth creation cost from $O(S\log(S))$ to $O(S^2)$ (S = subgraph G' size). Yet, as Sec. 3.1 explains, this extra cost is amortized for larger graphs as it is largely due to the fixed ground-truth training cost. More importantly, for any ground-truth technique used, our method has the same *linear* complexity in the input graph size as NNP-NET.

Our results show that we can create layouts which focus on the graph data, attribute data, or a mix of both, by changing a single parameter, even for hundreds of attributes per node. Our results match the results expected when using only graph data or only attributes for datasets for which we have such ground-truth layouts. The smooth change between these two layout extremes – desirable when users manipulate α – is confirmed by both visuals and the evolution of graph- and attribute stress values.

We see several immediate future work directions. Accelerating the ground-truth computation by using a faster t-SNE implementation (*e.g.* GPU-based [CRHC18]) would make the overhead of using attributes far smaller. Extending our method to handle time-dependent and multivariate graphs, a known challenge in graph drawing, should be relatively direct. Finally, using graph information, computed from attributes, to actually create better projections, *e.g.*, in terms of class visual separation, is a different way in which our method, that combines DR and graph drawing, can be useful.

References

[BP07] BRANDES U., PICH C.: Eigensolver methods for progressive multidimensional scaling of large data. In *Proc. Graph Drawing* (2007), Springer, pp. 42–53. doi:10.1007/978-3-540-70904-6_6. 1

[BTT22] BREDIUS C., TIAN Z., TELEA A.: Visual exploration of neural network projection stability. In *Proc. MLVis* (2022). 4

[CRHC18] CHAN D. M., RAO R., HUANG F., CANNY J. F.: T-SNE-CUDA: GPU-accelerated t-SNE and its applications to modern data. In *Proc. SBAC-PAD* (2018). doi:10.1109/CAHPC.2018.8645912. 4

[Den12] DENG L.: The MNIST database of handwritten digit images for machine learning research. *IEEE Signal Prog Mag 29*, 6 (2012), 141–142. doi:10.1109/MSP.2012.2211477. 2

[DH11] DAVIS T. A., HU Y.: The University of Florida sparse matrix collection. *ACM Trans Math Softw 38*, 1 (2011), 1–25. doi:10.1145/2049662.2049663. 3

[EHST20] ESPADOTO M., HIRATA N., SUMIKO T., TELEA A. C.: Deep learning multidimensional projections. *Inf Vis 19*, 3 (2020), 247–269. doi:10.1177/1473871620909485. 1

[FR91] FRUCHTERMAN T. M. J., REINGOLD E. M.: Graph drawing by force-directed placement. *Software – Practice and Experience 21*, 11 (1991), 1129–1164. doi:10.1002/spe.4380211102. 1

[GFV13] GIBSON H., FAITH J., VICKERS P.: A survey of two-dimensional graph layout techniques for information visualisation. *Inf Vis 12*, 3-4 (2013), 324–357. doi:10.1177/1473871612455749. 1

[GLA*21] GIOVANNANGELI L., LALANNE F., AUBER D., GIOT R., BOURQUI R.: Deep neural network for drawing networks, $(DNN)^2$. In *Graph Drawing* (2021), Springer, pp. 375–390. doi:10.1007/978-3-030-92931-2_27. 1

[GMVW24] GRÖTSCHLA F., MATHYS J., VERES R., WATTENHOFER R.: CoRe-GD: A Hierarchical Framework for Scalable Graph Visualization with GNNs. In *Proc. ICLR* (2024). doi:10.48550/arXiv.2402.06706. 1

[GV17] GIBSON H., VICKERS P.: graphTPP: A multivariate based method for interactive graph layout and analysis. *arXiv:1712.05644* (2017). doi:10.48550/arXiv.1712.05644. 1

[HGT26] HARTSKEERL I., GRÖTSCHLA F., TELEA A.: NNP-NET for multivariate graphs source code, 2026. https://github.com/IlanHartskeerl/NNP-NET. 3

[HMvW*25] HARTSKEERL I., MCHEDLIDZE T., VAN WAGENINGEN S., VANGORP P., TELEA A.: NNP-NET: Accelerating t-SNE graph drawing for very large graphs by neural networks. In *Proc. Graph Drawing* (2025), pp. 22–1. doi:10.4230/LIPIcs.GD.2025.22. 1, 2, 3

[HYL17] HAMILTON W., YING Z., LESKOVEC J.: Inductive representation learning on large graphs. *Advances in neural information processing systems 30* (2017). 2

[Kri09] KRIZHEVSKY A.: Learning multiple layers of features from tiny images, 2009. Univ. Toronto, Canada. 2

[KRM*17] KRUIGER J. F., RAUBER P. E., MARTINS R. M., KERREN A., KOBOUROV S., TELEA A. C.: Graph layouts by t-SNE. *Compu Graph Forum 36*, 3 (2017), 283–294. doi:10.1111/cgf.13187. 1, 3

[RFT17] RAUBER P., FALCÃO A., TELEA A.: Projections as visual aids for classification system design. *Inf Vis 17*, 4 (2017), 282–305. doi:10.1177/1473871617713337. 2

[RKH*21] RADFORD A., KIM J. W., HALLACY C., RAMESH A., GOH G., AGARWAL S., SASTRY G., ASKELL A., MISHKIN P., CLARK J., ET AL.: Learning transferable visual models from natural language supervision. In *Proc. ICML* (2021), pp. 8748–8763. 2

[STSW24] SHEN L., TAI Z., SHEN E., WANG J.: Graph exploration with embedding-guided layouts. *IEEE TVCG 30*, 7 (2024). doi:10.1109/TVCG.2023.3238909. 1

[TCG22] TIEZZI M., CIRAVEGNA G., GORI M.: Graph neural networks for graph drawing. *IEEE Trans. on Neural Networks and Learning Systems 35*, 4 (2022), 4668–4681. doi:10.1109/TNNLS.2022.3184967. 1

[Tor52] TORGERSON W. S.: Multidimensional scaling: I. Theory and method. *Psychometrika 17*, 4 (1952), 401–419. doi:10.1007/BF02288916. 1

[VBW17] VEHLOW C., BECK F., WEISKOPF D.: Visualizing group structures in graphs: A survey. *Comp Graph Forum 36*, 6 (2017), 201–225. doi:10.1111/cgf.12872. 1

[vdM14] VAN DER MAATEN L.: Accelerating t-SNE using tree-based algorithms. *J Mach Learn Res 15*, 1 (2014), 3221–3245. doi:10.5555/2627435.2697068. 3

[vdMH08] VAN DER MAATEN L., HINTON G.: Visualizing data using t-SNE. *J Mach Learn Res 9*, 11 (2008). 1, 3

[XRV17] XIAO H., RASUL K., VOLLGRAF R.: Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *arXiv:1708.07747* (2017). doi:10.48550/arXiv.1708.07747. 2

[YCS16] YANG Z., COHEN W., SALAKHUDINOV R.: Revisiting semi-supervised learning with graph embeddings. In *Proc. ICML* (2016), pp. 40–48. 2