Abstract—We introduce a nonlinear method for directly embedding large, sparse, stochastic graphs into low-dimensional spaces, without requiring vertex features to reside in, or be transformed into, a metric space. Graph data and models are prevalent in real-world applications. Direct graph embedding is fundamental to many graph analysis tasks, in addition to graph visualization. We name the novel approach SG-t-SNE, as it is inspired by and builds upon the core principle of t-SNE, a widely used method for nonlinear dimensionality reduction and data visualization. We also introduce t-SNE-II, a high-performance software for 2D, 3D embedding of large sparse graphs on personal computers with superior efficiency. It empowers SG-t-SNE with modern computing techniques for exploiting in tandem both matrix structures and memory architectures. We present elucidating embedding results on one synthetic graph and four real-world networks. More experimental results and comparisons are in Supplementary Material.1

I. INTRODUCTION

Big and sparse graph or network data exist and emerge in various research fields and real-world applications. Relational data of this type are in great diversity, such as biological networks, social, friend or co-author networks, commercial product networks, food webs, ecological networks, telecommunication networks, word co-occurrence networks, image-net, word2vec product networks, word co-occurrence networks, image-net, word2vec networks, word co-occurrence networks, image-net, word2vec [1]–[15]. Graph/network analysis plays an important role in data analysis in general.

A graph $G(V, E)$ has a set $V$ of vertices and a set $E$ of edges. The vertices (nodes) are an abstraction of entities or objects that have concrete forms or possess particular attributes in a real-world context. The vertices may stand for molecules, proteins, neuron cells, species, products, customers, words, documents, signals or time series, images or pixels. An edge (link) connecting two vertices represents a certain relationship, an interaction, or proximity between them. Additional graph information may include vertex attributes and edge weights.

Fundamental to many graph analysis tasks, graph embedding renders a mapping from vertices to their coded vectors (abstract features) in a code space (the embedding space), with code length $L$. The mapping is subject to one or more conditions to preserve (or reconstruct) certain graph properties in the embedding space. If the edge weights indicate pairwise (geometric or geodesic) distances between nodes, one wishes to preserve the pairwise distances as much as possible. This type of graph embedding is vertex embedding. Often, the code space for large graphs resides in $\mathbb{R}^L$, such as with word2vec and its variants [16], [17]. Various graph analysis tasks are subsequently carried out in an embedding space. They include (semi-, un-) supervised classification or stratification, abnormality detection, noise reduction, propagation patterns, content recommendation and other types of prediction. The feature vector length $L$ is chosen sufficiently large for these multiple analytical tasks. In recent years, numerous types of high-dimensional graph embedding techniques are tried out with neural networks [18]–[28].

High-dimensional graph embedding is often followed and accompanied by dimension reduction and low-dimensional embedding for multiple purposes: (i) visual assessment, inspection and summary of the analysis results in large quantities; (ii) facilitating interactive exploration; and (iii) making connections to the original, interpretable attributes. There is a plethora of low-dimensional graph embedding techniques [29]–[32]. Most are distance-based, preserving geometric neighborhood, assuming vertex features in a metric space.

We make two major contributions in the present work. First, we introduce a novel nonlinear approach for directly embedding large, sparse, stochastic graphs into low-dimensional spaces, without requiring vertex features to reside in, or be transformed into, a metric space. There are a lot of stochastic graphs, in real-world applications, without feature vectors readily in a metric space or available for public use. Our approach, named SG-t-SNE, is inspired by and builds upon the...
core principle of t-SNE, which is widely used for nonlinear dimensionality reduction and data visualization. There are many variants of t-SNE [33]–[40]. In SG-t-SNE, by a radical removal of the existing restrictions, we extend the use of t-SNE to the entire realm of stochastic graphs, no longer limited to distance-based $k$-nearest neighbor ($k$NN) graphs. SG-t-SNE is also equipped with a parameterized non-linear rescaling mechanism. It explores the sparsity to a much greater scope. In the case of a $k$NN graph, SG-t-SNE is capable of rendering much better embedding results than conventional t-SNE. See experimental comparisons in Supplementary Material.

Secondly, we introduce a software t-SNE-II. It accelerates 2D graph embedding up to $5 \times$ in execution time in comparison to the best among the existing t-SNE variants. More importantly, it enables 3D embedding of large sparse graphs on personal computers, with superior efficiency. We demonstrate in Fig. 1 that 3D embedding has greater capacity of preserving local and structural connectivity. Steerable visualization of all 3D embedding results is available at the online Supplementary Material. We present also experimental results on two biological networks, a social network created by Google, and an Amazon product network. The embedding results are characteristic and elucidating.

II. EMBEDDING SPARSE STOCHASTIC GRAPHS

We introduce SG-t-SNE, with which we apply and extend the essential principle of t-SNE to the entire realm of sparse stochastic graphs. A stochastic graph $\mathcal{G} = (V, E, P)$ is associated with a stochastic weight matrix $P = [p_{ij}]$. At each vertex $i$, $\sum_{j} a_{ij} p_{ji} = 1$, where $A = [a_{ij}]$ is the binary-valued adjacency matrix (the sparsity mask) of the graph. We assume without loss of generality that $\mathcal{G}$ has no isolated (0-degree) vertices. We are especially concerned with real-world graphs that have a large number of vertices and sparse connections ($|E| = O(|V|)$. Real-world graphs do not necessarily have constant degrees.

A. An investigative review of t-SNE

We describe first the essence of t-SNE. Provided with a graph $\mathcal{G}$ with stochastic weights at each vertex, described by a stochastic matrix $\mathbf{P}_c = (P_{cij})$, and given a prescribed modest dimensionality $d$, t-SNE generates vertex embedding coordinates $\mathcal{Y}_d^* = \{\mathbf{y}\}_{i=1}^n \in \mathbb{R}^d$. With its configuration specified by the Student t-distribution $Q(\mathcal{Y}) = \{q_{ij}\}$,

$$q_{ij} = \frac{(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}}{\sum_k (1 + \|\mathbf{y}_k - \mathbf{y}_j\|^2)^{-1}}, \quad i \neq j,$$

(1)

the embedding arrives at the optimal matching to the joint distribution $\mathbf{P} = (\mathbf{P}_c + \mathbf{P}_c^T)/2n$ in the sense that

$$\mathcal{Y}_d^* = \arg \min_{\mathcal{Y}} \text{KL}(\mathbf{P} \| Q(\mathcal{Y})),
$$

(2)

where $\text{KL}$ denotes the Kullback-Leibler divergence.

The standard version of t-SNE, however, is developed, used, viewed and reviewed as a nonlinear dimension reduction algorithm, due primarily to the following interface to the distribution matching objective (2). The input is a set of data points

in terms of feature vectors $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^n$ in an $L$-dimensional space $\mathbb{R}^L$ endowed with a distance function $d(\cdot, \cdot)$. The feature vector length $L$ is (much) larger than the embedding dimension $d$. The algorithm locates the $k$ nearest neighbors ($k$NN) of each vertex, and integrates them into the $k$NN graph, which is regular (with constant degree $k$). Let $\mathbf{A}$ be the binary-valued adjacency matrix. Denote by $\mathbf{D}$ the matrix of squared pairwise distances, $\mathbf{D} = [d_{ij}^2] = [d^2(x_i, x_j)]$. Then, the distance-weighted $k$NN matrix $\mathbf{D} \odot \mathbf{A}$ (Hadamard product) is converted to a stochastic matrix $\mathbf{P}_c$ by parameterized exponentiation and normalization,

$$p_{ij}(\sigma_i) = \frac{a_{ij} \exp(-d_{ij}^2/2\sigma_i^2)}{\sum_i a_{ij} \exp(-d_{ij}^2/2\sigma_i^2)}. \quad (3)$$

For every vertex $i$, the Gaussian parameter $\sigma_i$ is determined by the equation

$$-\sum_j a_{ij} p_{ji}(\sigma_i) \log(p_{ji}(\sigma_i)) = \log(u), \quad (4)$$

where the prescribed perplexity $u$ is a hyper-parameter. In fact, equation (4) equalizes all vertices by the conditional entropies.

We make a couple of remarks on the relationship between $u$ and $k$. Commonly, the heuristic rule $k = [3u]$ is used. Veritably, the condition $0 < u < k$ is necessary for the solutions to (4) to exist at all vertices. With a larger perplexity $u$, one would increase $k$ proportionally to get a better embedding in the sense of (2), at the expense of a denser graph, larger memory requirement and longer execution time.

To unleash the greater potential with the essential t-SNE, we must unravel two elaborate restraining knots in the standard form. One is the coupling between distance and stochastics by (3). The other is the pairing between vertex degree and perplexity by (4).
B. Advantages of admitting stochastic graphs

Stochastic graph data arise in numerous and various situations. Embedding of stochastic graphs will benefit network analysis and network-based data analysis in general.

The stochastic kNN graphs obtained by (3) and (4) are only a subset of stochastic graphs. While they play a key role for t-SNE being seen and used as a tool for nonlinear dimensionality reduction and data visualization on the one hand, they obscure the essential principle of t-SNE in several ways on the other hand. (i) Real-world graph data stem from various sources [1]–[4], [6]–[12], [41]. Often, the native and interpretable descriptors of the entity vertices are not readily in a metric space. The descriptors may contain both numerical and categorical data attributes. For such data, t-SNE may be preceded by a high-dimensional metric-space embedding, using machine learning tools (e.g., neural networks) and techniques [16], [17], as in parametrized t-SNE [42], at a very high computational cost. (ii) In the case where the feature descriptors are in a metric space, constructing the kNN graph is computationally expensive. Randomized methods are used to reduce the asymptotic complexity to \(O(n \log n)\) [43], [44]. However, such methods do not leverage domain-specific spaces of random variables [45]. (iii) The kNN graph and the outcome of t-SNE vary with the choice of vector space, distance function, and kNN search methods. (iv) The stochastic recasting of a kNN graph is exclusively confined in conventional t-SNE to the conditional Gaussian distribution, not inclusive of other probabilistic models. The above limitations are common to many t-SNE variants [33]–[40] and other multi-dimensional scaling (MDS) algorithms [29]–[31]. There exist nonlinear dimension reduction methods that attempt to circumvent the kNN search with certain sampling strategies [32].

Stochastic graphs are found or obtained in a multitudine of other ways. We give two examples. First, for high-security information networks, the vertex content features are not publicly accessible and their links are stochastically encrypted [46]. Second, network data are often available only with binary-valued links. Nonetheless, the stochastic weights of the links may be recovered to significant extent from the local neighborhood topologies and the global network topology. We use in this paper two such stochastic graphs from a recent study [47]: one is of a social network, the other of an e-commerce network. In addition, real-world graphs are typically irregular, with high-degree hub nodes and 1-degree leaf nodes. See for example the degree distribution of the social network in Fig. 3.

In short, with stochastic graphs, we effectively create a clear abstraction of distinguishing graph embedding from graph construction. We also establish an interface between the two key processes, disentangling them for integrating the best in each. We explore the deep potential of the t-SNE essence.

C. Introduction of SG-t-SNE

Our algorithm SG-t-SNE follows the essential principle of t-SNE and removes the barriers in its conventional version. SG-t-SNE admits any stochastic graph \(\mathcal{G} = (V, E, P_c)\), including stochastic kNN graphs. SG-t-SNE is capable of not only adapting to the sparse pattern in the input, but also exploiting the pattern for better distribution matching and vertex embedding with a non-linear re-scaling mechanism.

Specifically, we generate from \(P_c\) a parameterized family of stochastic matrices, \(P_c(\lambda)\), via nonlinear rescaling. The role of the rescaling mechanism is to explore and exploit the potential in distribution matching (2) from the \(P\) side, given the fixed t-distribution (a Cauchy distribution) on the \(Q\) side. Unlike the perplexity \(u\) in the conventional t-SNE, the parameter \(\lambda\) for SG-t-SNE exploits the sparsity without imposing any
constraint. In the special case of a weighted kNN graph, $\lambda$ is untangled from $k$.

We describe the mechanism and effect of nonlinear rescaling with a single parameter $\lambda$. Let $\phi(\cdot)$ be a rescaling kernel function that is monotonically increasing over $\mathbb{R}_+$, with $\phi(0) \geq 0$. For any $\lambda > 0$, SG-t-SNE generates $P_c(\lambda)$ as follows. Let $A$ be the binary-valued adjacency matrix (i.e., the sparsity mask) of graph $G$, which is invariant to any change in $\phi$ or $\lambda$. We determine the rescaling exponent $\gamma_i$ at each vertex $i$ by the equation

$$\sum_j a_{ij} \phi(p_{ij}^{\gamma_i}) = \lambda,$$

and then reshape and rescale the conditional probability

$$p_{ij}(\lambda) = \frac{a_{ij} \phi(p_{ij}^{\gamma_i})}{\lambda}.$$

Unlike (4), the rescaling solutions exist unconditionally.

The rescaling mechanism introduces an additional degree of freedom to exploit the sparsity in an arbitrary form. In particular, we use the identity function $\phi(x) = x$ as the rescaling kernel for the experiments presented in this paper. At $\lambda = 1$ we get $\gamma_i = 1$ and $P_c(1) = P_c$, the input matrix. At an integer value $\lambda = k > 1$, if degree($\ell$) = $k$, then $\gamma_\ell = 0$ and $p_{j|\ell} = 1/k$. In the special case of a regular graph with degree $k$, if we set $\lambda = k$, then $\gamma_i = 0$ at all vertices, i.e., $P_c(k) = A/k$. In general, at $\lambda \neq 1$, the ratio between every nonzero element to the peak element ($p_{j|i}/\max_i p_{j|i}$) is changed to ($p_{j|i}/\max_i p_{j|i}$)$^{\gamma_i}$. The weight distribution for each vertex is re-shaped, according to the re-scaling equation (5), in autonomous adaptation to the local sparsity.

In implementation, only two changes are made from the conventional version. One is at the interface, which takes either a distance-based kNN graph as in the conventional t-SNE or a stochastic matrix directly. The other is the change from the perplexity equation (4) to the rescaling equation (6).

We show in Figs. 2 and 3 the embeddings of two real-world networks, enabled by SG-t-SNE. Each exhibits a distinctive cluster structure. We show in Fig. 5 that even with kNN graphs at input, the embedding by t-SNE at $k = 150$, $u = 50$ is matched or outperformed by that with SG+t-SNE at $k = 90$, $\lambda = 10$, on a sparser matrix and at a lower computation cost. We recommend multiple views at various values of $\lambda$.

### III. Rapid Spaceland Embedding

The practical use of t-SNE is largely confined to Flatland (2D) or Lineland (1D) embedding, despite the prototype implementation by van der Maaten\(^2\) supporting embedding in several dimensions. We advocate Spaceland (3D or higher) embedding. Among abundant evidence, we illustrate in Figs. 1, 3, and 4, the extended capacity to capture and reveal multifold connections between (overlapping) subpopulations. With Spaceland embedding, we endorse “the enlargement of the imagination” [50].

There were multiple obstacles to Spaceland embedding. The journey of t-SNE is marked by continuous efforts and progress in reducing the arithmetic complexity and execution time of searching for an optimal embedding. We present t-SNE-II, a software renovation, a new milestone in performance. It enables rapid Spaceland embedding of large sparse graphs on modern desktop or laptop computers, which are available and affordable to researchers by and large.

The embedding coordinates, i.e., $d_n$ parameters, are determined by t-SNE via applying the gradient descent method to the minimization problem (2). The computation per iteration step is dominated by the calculation of the gradient, which is reformulated in two terms by van der Maaten [35].

$$\frac{\partial (KL)}{\partial y_i} = \frac{4}{Z} \sum_{i \neq j} p_{ij} q_{ij} (y_i - y_j) - \frac{4}{Z} \sum_{i \neq j} q_{ij}^2 (y_i - y_j),$$

\(^2\)https://fvdmaaten.github.io/tsne
where \( Z = \sum_{i,j} q_{ij} \). The terms are named by analogy to the attractive and repulsive forces in molecular dynamics simulations. The attractive term can be cast as the sum of multiple matrix-vector products with a sparse matrix \( PQ = [p_{ij} q_{ij}] \) and \( d \) coordinate vectors, one along each embedding dimension. Similarly, the repulsive term can be cast as the sum of multiple matrix-vector products with a dense matrix \( QQ = [q_{ij} q_{ij}] \) and \( d \) coordinate vectors. The calculation of the attractive term takes \( O(dm) \) arithmetic operations, where \( m = |E| \) is the number of edges. A naive calculation of the repulsive term takes \( O(dn^2) \) arithmetic operations. This quadratic complexity would prohibit the use of t-SNE with large graphs.

### A. Fast approximations to the repulsive term

Two existing t-SNE variants use approximate algorithms to calculate the repulsive term. They reduce the quadratic complexity to \( O(c^2(e^{-1}) n \log(n)) \), where \( c(e^{-1}) > 2 \) increases reciprocally with \( e \), a prescribed approximation error tolerance. The specific value of \( c \) in the prefactor is determined also on the choice of approximation algorithm and its parameters. The first approximation algorithm, by van der Maaten, adopts the tree code based on the Barnes-Hut (BH) algorithm [51]. We refer to this version as t-SNE-BH. An alternative approximation algorithm, named Flt-SNE, emerged last year [40]. We describe the fundamental concepts and distinctive approaches behind the two approximation algorithms, instead of reviewing each in detail. The matrix-vector multiplication with \( QQ \) is a convolution with the Cauchy kernel on non-equispaced, scattered data points \( \{y_i\}_{i=1}^{n} \). We refer to such a convolution as “nuConv”. A convolution may have broad or narrow (windowed) support. Narrow-support convolution has asymptotic arithmetic complexity \( O(n \log(n)) \), where \( w \) is the bandwidth or support window size; the corresponding matrix is sparse and banded. The \( \text{nuConv} \) with \( QQ \) is of broad support and dense. Fortunately, the matrix structure can be exploited.

There are two renowned and distinctive families of algorithms for fast \( \text{nuConv} \) of broad support. One family is based on the fast multipole method (FMM), with asymptotic arithmetic complexity \( O(n \log(n)) \) [52]–[54]. FMM splits a convolution of broad support into \( O(\log(n)) \) convolutions of narrow supports at multiple spatial levels. Its extended family includes the Barnes-Hut algorithm, with complexity \( O(n \log(n)) \) [51]. The other family is based on non-uniform fast Fourier transforms (nuFFTs), with arithmetic complexity \( O(n \log(n)) \) [21], [55]. Flt-SNE takes the latter approach.

Flt-SNE is available for 1D/2D embedding only. By a careful comparison, its arithmetic complexity is lower than t-SNE-BH at the same accuracy level. Nonetheless, the execution time for 2D embedding with Flt-SNE (or t-SNE-BH) is now dominated by the computation of the attractive term; see Fig. 6. Furthermore, both Flt-SNE and t-SNE-BH face a steep rise in execution time for 3D embedding. Flt-SNE also suffers from exponential growth in memory demand due to zero padding for explicit conversion to periodic convolution.

### B. Introduction of t-SNE-II

With t-SNE-II, we advance the entire gradient calculation of (7) and enable Spaceland (3D) embedding in shorter time than Flatland (2D) embedding with Flt-SNE or t-SNE-BH. We overcome challenges at multiple algorithmic stages with innovative approaches. We utilize the matrix structure and the memory architecture in concert.

We not only expedite further the repulsive term computation with dense matrix \( QQ \), but also accelerate the attractive term computation with sparse matrix \( PQ \). Furthermore, we make swift translocation of \( \{y_i\} \) between the two terms.

**Fast computation with \( QQ \):** We take the spectral approach, similar to Flt-SNE. Set first an equispaced grid \( G \) in the spatial embedding domain. The \( \text{nuConv} \) with \( QQ \) is then factored into three consecutive convolutional operations which we code-name as S2G, G2G, and, G2S. The S2G and G2S convolutions translate \( \{y_i\} \) to their neighboring grid points, and vice versa. G2S is instrumented as a local interpolation and S2G is its transpose. Both are of narrow support, with arithmetic complexity \( O(w^d n) \), where \( w \) is the interpolation window size per side. Factor G2G is a non-periodic convolution across the grid \( G \).

We make algorithmic innovations for each of the factor convolutions. Instead of explicit embedding G2G into a circulant one as in Flt-SNE, we use an implicit approach without augmenting the grid size and its memory usage by a factor...
of $2^d$, while maintaining the same arithmetic complexity. This memory saving allows a laptop or desktop computer to admit larger graphs. The factors S2G and G2S prolong the execution time due to irregular data access, a typical challenge in sparse matrix calculation. We introduce a dual grid $G_{\text{dual}}$ for gridding the scattered data points $\{y_i\}$. Prior to S2G, the scattered points are binned into the dual-grid cells. Then, S2G translates the gridded data on $G_{\text{dual}}$ to grid $G$, on which $G_{2G}$ takes place. This approach significantly improves the data locality and parallelism scope in S2G, and shortens the S2G time, accounting for the gridding overhead. The same holds true for G2S. We omit implementation details.

**Fast computation with PQ:** Irregular sparse pattern of PQ invokes irregular memory accesses which incur long latency on modern computers with hierarchical memories [57]. The execution time of FIt-SNE is evidently dominated by the computation with PQ at $k = 90$ and $u = 30$; see Fig. 6. The CPU is frequently in data-hungry state, the computation speed is limited by the memory bandwidth.

We mitigate the problem with a preprocessing step, based on the fact that matrix PQ inherits the fixed sparsity of matrix $P$, despite the dense structure and dynamic change of $Q$. We permute $P$ such that similar rows and columns are clustered and placed together. The permuted matrix becomes block-sparse with denser blocks (BSDB), not necessarily block diagonal. A denser block implies higher data locality in memory reads and writes. We employ several efficient ways for clustering rows and columns (symmetrically). For a $k$NN graph at input, in particular, t-SNE-II has options among PCA-based cluster analysis and the SD-DP classification analysis [56]. The permutation overhead is amortized across all iterations. It is well paid off; see Fig. 6.

### Table I: Architecture specification of three multi-core computers used for the embedding experiments. Memory bandwidth is measured with the parallel STREAM copy benchmark [60].

| CPU                      | Clock | Cores | L1  | L2  | L3  | RAM   | BW   |
|--------------------------|-------|-------|-----|-----|-----|-------|------|
| Intel Core i7-4558U      | 2.80  | 2     | 64  | 256 | 4   | 8     | 10.5 |
| Core i7-6700             | 3.40  | 4     | 32  | 256 | 8   | 32    | 19.9 |
| Xeon E5-2640v4*          | 2.40  | 10    | 32  | 256 | 25  | 256   | 30.7 |

*2 sockets (non-uniform memory access—NUMA)

In implementation detail, we adopt the storage format and sparse matrix computation routines provided in the Compressed Sparse Blocks (CSB) library [58], [59].

**Fast data translocation:** The embedding coordinate data $\{y_i\}$ undergo dynamic change during the process of iterative search and matching (2). They are accessed in the computation with both QQ and PQ, but in different orderings. Matrix $P$, and hence PQ, is permuted, once for all, by row/column clusters. The order $\{y_i\}$ is accessed in the computation with QQ, is determined by the dual-grid cells for S2G and G2S. At each iteration step, we translocate the data $\{y_i\}$, back and forth between the two terms, in multiple stages. In essence, a point-to-point cross permutation is factored into layers of block-to-block permutations.

We use II in the name of t-SNE-II to signify the role of permutations, factored permutation and the rational behind. Permutations are unaccounted for in arithmetic complexity. They play, however, the key role of making the algorithm and its building blocks attuned to the memory hierarchy. Without this harmonious algorithm-architecture mapping, memory access latency typically dominates the execution time in computation with large, sparse or compressible matrices [61], [62].

### IV. Additional remarks

The embedding results obtained by SG-t-SNE, powered by t-SNE-II, are elucidating. The embedding results with orkut and Amazon show clearly distinctive cluster or communites patterns. Furthermore, we find overlapping and non-overlapping regions between communities in orkut. The embedding of Amazon may serve as a map for market study and analysis, in assessing previous claimed product clusters and making new hypotheses or predictions. Even with $k$NN graphs, 3D embedding liberates us from 2D confinement and takes us to a deeper space where we can see multiple views at once, discover previously unseen connections and separations, such as with the PBMCs data and the E18-MBCs data. Additionally, we bring out the fascinating beauty of orkut and Amazon in their respective embedding spaces.

### Acknowledgments

We thank Tiancheng Liu for providing the sparse and stochastic graphs for the orkut and Amazon networks. We thank Pan tắtis Vouzaksakis and Xenofon Theodoridis for their assistance with the experiments.
[42] L. van der Maaten, “Learning a parametric embedding by preserving local structure,” in Proceedings of the 12th International Conference on Artificial Intelligence and Statistics, vol. 5, 2009, pp. 384–391.

[43] M. Muja and D. G. Lowe, “Scalable nearest neighbor algorithms for high dimensional data,” IEEE TPAMI, vol. 36, no. 11, pp. 2277–2240, 2014.

[44] ——, “Fast approximate nearest neighbors with automatic algorithm configuration,” in International Conference on Computer Vision Theory and Application. INSTICC Press, 2009, pp. 331–340.

[45] T. U. Consortium, “Reorganizing the protein space at the Universal Protein Resource (UniProt),” Nucleic Acids Research, vol. 40, no. D1, pp. D71–D75, 2011.

[46] M. Su, H. Zhang, X. Duy, and Q. Dai, “A novel stochastic-encryption-based P2P digital rights management scheme,” in IEEE International Conference on Communications, 2015, pp. 5541–5545.

[47] T. Liu and X. Sun, “Density-based detection of community hierarchy from graph description,” 2019, manuscript in preparation.

[48] “Transcriptional profiling of 1.3 million brain cells with the Chromium Single Cell 3’ solution,” 10x Genomics LIT000015 Chromium Million Brain Cells Application Note, 2017.

[49] G. X. Y. Zheng, J. M. Terry, P. Belgrader, P. Ryvkin, Z. W. Bent, R. Wilson, S. B. Ziraldo, T. D. Wheeler, G. P. McDermott, J. Zhu, M. T. Gregory, J. Shuga, L. Montesclaros, J. G. Underwood, D. A. Masquelier, S. Y. Nishimura, M. Schnall-Levin, P. W. Wyatt, C. M. Hindson, R. Bharadwaj, A. Wong, K. D. Ness, L. W. Beppu, H. J. Deeg, C. McFarland, K. R. Loeb, W. J. Valente, N. G. Ericson, E. A. Stevens, J. P. Radich, T. S. Mikkelsen, B. J. Hindson, and J. H. Bielas, “Massively parallel digital transcriptional profiling of single cells,” Nature Communications, vol. 8, p. 14049, 2017.

[50] E. A. Abbott, Flatland: A Romance of Many Dimensions. London, UK: Seeley & Co, 1884.

[51] J. Barnes and P. Hut, “A hierarchical O(N log N) force-calculation algorithm,” Nature, vol. 324, no. 6096, pp. 446–449, 1986.

[52] A. Yzelman and R. H. Bisseling, “Cache-oblivious sparse matrix–vector multiplication by using sparse matrix partitioning methods,” SIAM Journal on Scientific Computing, vol. 31, no. 4, pp. 3128–3154, 2009.