

Scalable Dimension Reduction for Large Abstract Data Visualization

Seung-Hee Bae, Judy Qiu, Geoffrey Fox
School of Informatics and Computing
Indiana University
Bloomington, IN
{sebae, xqiu, gcf}@cs.indiana.edu

Abstract—The ability to browse vast amounts of scientific data is critical to facilitate science discovery. High performance Multidimensional Scaling (MDS) algorithm makes it a reality by reducing dimensions so that scientists can gain insight into data set from a 3D visualization space. As multidimensional scaling requires quadratics order of physical memory and computation, a major challenge is to design and implement parallel MDS algorithms that can run on multicore clusters for millions of data points. Bases on our early work of parallel SMACOF algorithm, the authors have developed an interpolated approach, majorizing interpolation MDS (MI-MDS). It utilizes the known mapping from a subset of given in-sample data to effectively reduce computational complexity with minor cost of approximation. MI-MDS makes it possible to process huge data set with modest amounts of computation and memory requirements. Our experimental results show that the quality of interpolated mapping is comparable to that of the original SMACOF in a million chemical compounds data, where we construct a configuration of over two-million out-of-sample data into target dimension space.

Keywords-Multidimensional Scaling; Parallelism; Interpolation;

I. INTRODUCTION

Due to the innovative advancement in science and technologies, the amount of data to be processed or analyzed is rapidly growing and it is already beyond the capacity of the most commodity hardware we are using nowadays. To keep up with such fast development, study for data-intensive scientific data analyses [1] has been already emerging in recent years. It is a challenge for various computing research communities, such as high-performance computing, database, and machine learning and data mining communities, how to deal with such large and high dimensional data in this data deluge era. Unless developed and implemented carefully to overcome such limits, techniques will face soon the limit of usability. Parallelism is not an optional technology any more but an essential factor for various data mining algorithms, including dimension reduction algorithms, by the result of enormous (and keep increasing) size of the data to be dealt by those algorithms.

Among many data mining areas, such as clustering, classification, and association rule mining, and so on, dimension reduction algorithms are used to visualize high-dimensional data or abstract data into a low-dimensional target space. Dimension reduction algorithms can be used for the data

visualization which can be applied to following purposes: (1) representing unknown data distribution structure in human-perceptible space with respect to the pairwise proximity or topology of the given data, (2) verifying a certain hypothesis or conclusion related to the given data by showing the distribution of the data, and (3) investigating relationship among the given data by spatial display. This task is also getting more difficult and challenged by the huge amount of the given data.

Among the known dimension reduction algorithms, such as Principal Component Analysis (PCA), Multidimensional Scaling (MDS) [2], [3], Generative Topographic Mapping (GTM) [4], and Self-Organizing Maps (SOM) [5], to name a few, multidimensional scaling (MDS) has been extensively studied and used in various real application area, such as biology [6], [7], stock market analysis [8], computational chemistry [9], and breast cancer diagnosis [10]. In this paper, we focus on MDS algorithm and we try to investigate two different methods for the achievement of the scalability: (1) how to parallelize the well-known MDS algorithm in order to deal with large-scale data via cluster systems, and (2) how to reduce the computational complexity and memory requirement to scale up to millions of points.

Section II briefly defines and introduces the MDS problem. In Section III, we describe how to parallelize an MDS algorithm, named SMACOF. The interpolation method for MDS problem, which reduces the computational complexity from $\mathcal{O}(N^2)$ to $\mathcal{O}(n(N - n))$, is illustrated in Section IV, followed by the conclusion section in Section V.

II. MULTIDIMENSIONAL SCALING (MDS)

Multidimensional scaling (MDS) [2], [3], [11] is a general term for techniques of constructing a mapping for generally high-dimensional data into a target dimension (typically low dimension) with respect to the given pairwise proximity information. Mostly, MDS is used for achieving dimension reduction to visualize high-dimensional or abstract data into Euclidean low-dimensional space, i.e. two-dimension or three-dimension.

Generally, the proximity information, which is represented as an $N \times N$ dissimilarity matrix ($\Delta = [\delta_{ij}]$), where N is the number of points (objects) and δ_{ij} is the dissimilarity between point i and j , is given for the MDS problem,

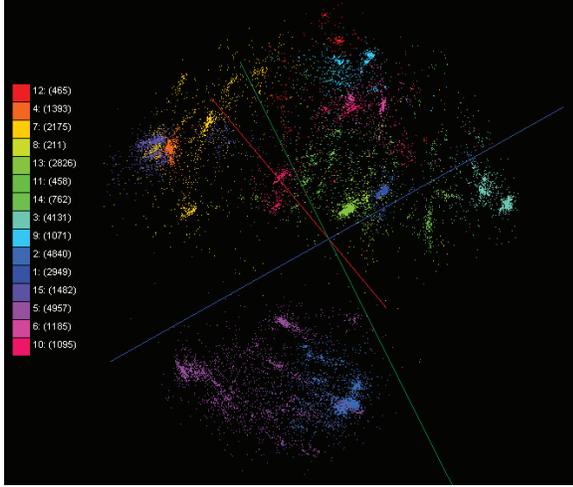


Figure 1. An example of data visualization of 30,000 biological sequences by an MDS algorithm, which is colored by a clustering algorithm

and the dissimilarity matrix (Δ) should agree with the following constraints: (1) symmetricity ($\delta_{ij} = \delta_{ji}$), (2) nonnegativity ($\delta_{ij} \geq 0$), and (3) zero diagonal elements ($\delta_{ii} = 0$). The objective of MDS techniques is to construct a configuration of the given high-dimensional data into low-dimensional Euclidean space, while each distance between a pair of points in the configuration is approximated to the corresponding dissimilarity value as much as possible. The output of MDS algorithms could be represented as an $N \times L$ configuration matrix \mathbf{X} , whose rows represent each data points x_i ($i = 1, \dots, N$) in L -dimensional space. It is quite straightforward to compute Euclidean distance between x_i and x_j in the configuration matrix \mathbf{X} , i.e. $d_{ij}(\mathbf{X}) = \|x_i - x_j\|$, and we are able to evaluate how well the given points are configured in the L -dimensional space by using suggested objective functions of MDS, called STRESS [12] or SSTRESS [13]. Definitions of STRESS (1) and SSTRESS (2) are following:

$$\sigma(\mathbf{X}) = \sum_{i < j \leq N} w_{ij} (d_{ij}(\mathbf{X}) - \delta_{ij})^2 \quad (1)$$

$$\sigma^2(\mathbf{X}) = \sum_{i < j \leq N} w_{ij} [(d_{ij}(\mathbf{X}))^2 - (\delta_{ij})^2]^2 \quad (2)$$

where $1 \leq i < j \leq N$ and w_{ij} is a weight value, so $w_{ij} \geq 0$.

As shown in the STRESS and SSTRESS functions, the MDS problems could be considered as a non-linear optimization problem, which minimizes STRESS or SSTRESS function in the process of configuring L -dimensional mapping of the high-dimensional data.

Fig. 1 is an example of data visualization of 30,000 biological sequence data, which is related to metagenomics study, by an MDS algorithm. The colors of the points in Fig. 1 represent the clusters of the data which is generated

by the pairwise clustering algorithm by deterministic annealing [14]. The data visualization in Fig. 1 shows the value of dimension reduction algorithms which produced lower dimensional mapping for the given data. We can see clearly the clusters without quantifying the quality of clustering methods statistically.

III. PARALLEL IMPLEMENTATION OF MDS

There are a lot of different algorithms to solve MDS problem, and Scaling by MAjorizing a COMplicated Function (SMACOF) [15], [16] is one of them. SMACOF is an iterative majorization algorithm to solve MDS problem with STRESS criterion. Although SMACOF has a tendency to find local minima due to its hill-climbing attribute, it is still a powerful method since the algorithm, theoretically, guarantees to decrease STRESS (σ) criterion monotonically. For the mathematical details of SMACOF algorithm, please refer to [3].

To parallelize SMACOF, it is essential to ensure load balanced data decomposition as much as possible. Load balance is important not only for memory distribution but also for computation distribution, since parallelization makes implicit benefit to computation as well as memory distribution, due to less computing per process. We decompose an $N \times N$ matrix to $m \times n$ block decomposition, where m is the number of block rows and n is the number of block columns, and the only constraint of the decomposition is $m \times n = p$, where $1 \leq m, n \leq p$. Thus, each process requires only approximately $1/p$ of full memory requirements of SMACOF algorithm. Fig. 2 illustrates how we decompose each $N \times N$ matrices with 6 processes and $m = 2, n = 3$. Without loss of generality, we assume $N \% m = N \% n = 0$ in Fig. 2.

For $N \times N$ matrices, such as $\Delta, \mathbf{V}^\dagger, \mathbf{B}(\mathbf{X}^{[k]})$, and so on, each block M_{ij} is assigned to the corresponding process P_{ij} , and for $\mathbf{X}^{[k]}$ and $\mathbf{X}^{[k-1]}$ matrices, $N \times L$ matrices where L is the target dimension, each process has full $N \times L$ matrices because these matrices are relatively much small size and it results in reducing a number of additional message passing routine calls. By scattering decomposed blocks to distributed memory, now we are able to run SMACOF with huge data set as much as distributed memory allows in the cost of message passing overheads and complicated implementation. The details of how to parallelize SMACOF algorithm is illustrated in [17].

In order to analyze the parallel scalability, we experiment the parallel SMACOF algorithm with real data set, which is obtained from PubChem database¹. We investigated the scalability of parallel SMACOF by running with different number of processes, e.g. $p = 64, 128, 192$, and 256. On the basis of the data decomposition experimental result in [17],

¹PubChem, <http://pubchem.ncbi.nlm.nih.gov/>

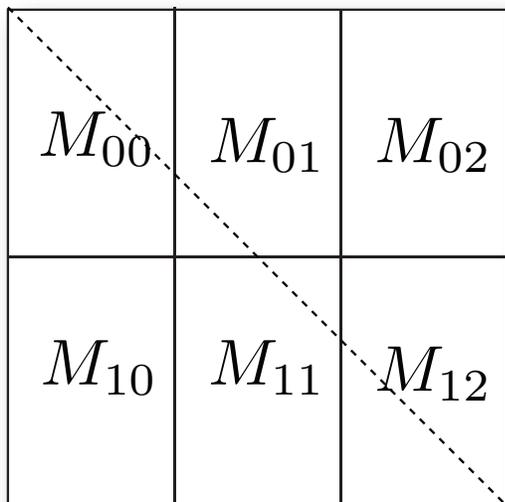


Figure 2. An example of an $N \times N$ matrix decomposition of parallel SMACOF with 6 processes and 2×3 block decomposition. Dashed line represents where diagonal elements are.

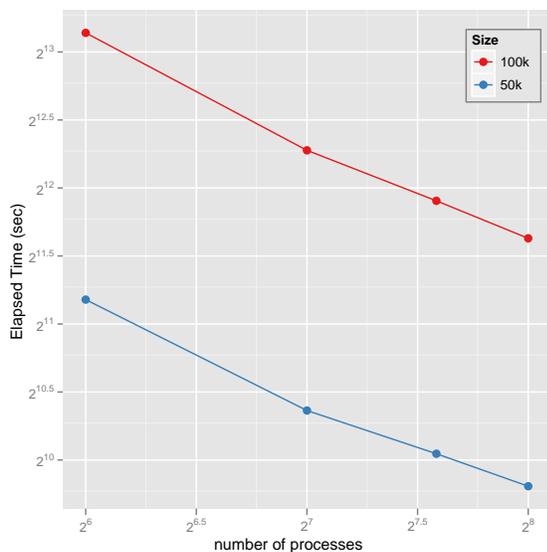


Figure 3. Runtime of parallel SMACOF for 50K and 100K PubChem data with respect to the number of processes. We choose balanced decomposition as much as possible, i.e. 8×8 for 64 processes. Note that both x and y axes are log-scaled.

the balanced decomposition has been applied to this process scaling experiments.

The elapsed time of parallel SMACOF with two large data sets, 50k and 100k, is shown in Fig. 3, and the corresponding relative efficiency of Fig. 3 is shown in Fig. 4. Note that both coordinates are log-scaled, in Fig. 3. As shown in Fig. 3, the parallel SMACOF achieved performance improvement as the number of parallel units (p) increases. However, the

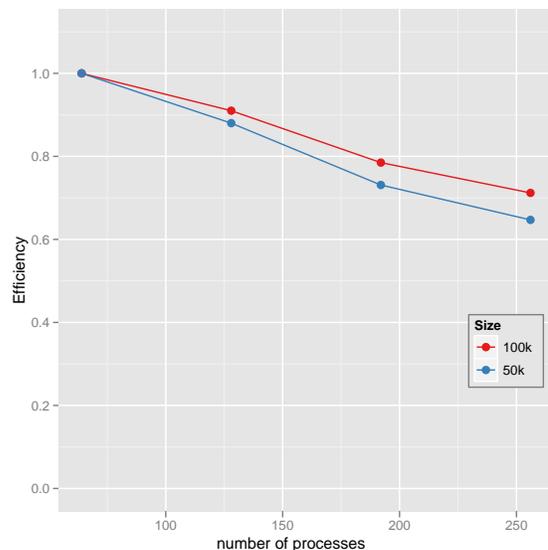


Figure 4. Efficiency of parallel SMACOF for 50K and 100K PubChem data with respect to the number of processes. We choose balanced decomposition as much as possible, i.e. 8×8 for 64 processes.

performance enhancement ratio (a.k.a. efficiency) is reduced as p increases, which is demonstrated in Fig. 4. The reason of reducing efficiency is that the ratio of message passing overhead over the assigned computation per each process is increased due to more message overhead and less computing portion per process as p increases.

IV. INTERPOLATION APPROACH TO MDS

As a result of distributed parallelization of SMACOF algorithm, we can configure a mappings of large high-dimensional data by utilizing cluster systems. We showed that we can generate a mapping of 100,000 pubChem data by parallel SMACOF in Section III through our cluster systems. However, the computational capability of the parallel SMACOF is still constrained by the $\mathcal{O}(N^2)$ memory and computation requirement, and soon it will meet the resource limitation as we keep increasing the data size. For instance, the tested cluster system in Section III has total 1.536 TB of main memory, but the SMACOF algorithm requires around 1.92 TB of main memory, which is larger than the total main memory of the cluster system, for running with 200,000 data.

To solve this obstacle, we developed a simple interpolation approach based on pre-mapped MDS result of the sample of the given data. Our interpolation algorithm is similar to k nearest neighbor (k -NN) classification [18], but we approximate new mapping position of the new point based on the positions of k -NN, among pre-mapped subset data, instead of classifying it. For the purpose of deciding new mapping position in relation to the k -NN positions, iterative majorization method is applied as in

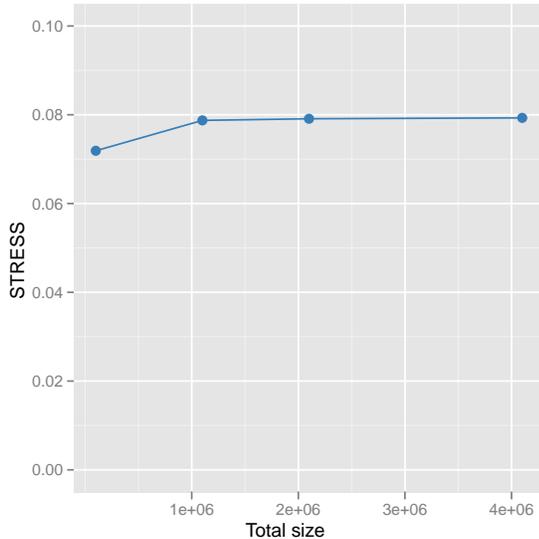


Figure 5. STRESS value change of Interpolation larger data, such as 1M, 2M, and 4M data points, with 100k sample data. The initial STRESS value of MDS result of 100k data is 0.0719.

Table I
LARGE-SCALE MI-MDS RUNNING TIME (SECONDS) WITH 100K
SAMPLE DATA

1 Million	2 Million	4 Million
731.1567	1449.1683	2895.3414

SMACOF [15], [16] algorithm. The details of mathematical majorization equations for the proposed out-of-sample MDS algorithm could be found in [19], and it is called Majorizing Interpolation MDS (hereafter *MI-MDS*).

Now, scalability of the proposed interpolation algorithm is examined as following: we fix the sample data size to 100k, and the interpolated data size is increased from one millions (1M) to two millions (2M) to four millions (4M). Then, the STRESS value is measured for each running result of total data, i.e. 1M + 100k, 2M + 100k, and 4M + 100k. The measured STRESS value is shown in Fig. 5. There are some quality lost between the full MDS running result with 100k data and the 1M interpolated results based on that 100k mapping, which is about 0.007 difference in normalized STRESS criteria. However, there is no much difference between the normalized STRESS value of the 1M, 2M, and 4M interpolated result, although the sample size is quite small portion of total data and the out-of-sample data size increases as quadruple. From the above result, we could consider that the proposed MI-MDS algorithm works well and scalable if we are given a good enough pre-configured result which represents well the structure of the given data. Note that it is not possible to run SMACOF algorithm with only 200k data points due to memory bound, within the same system we used.

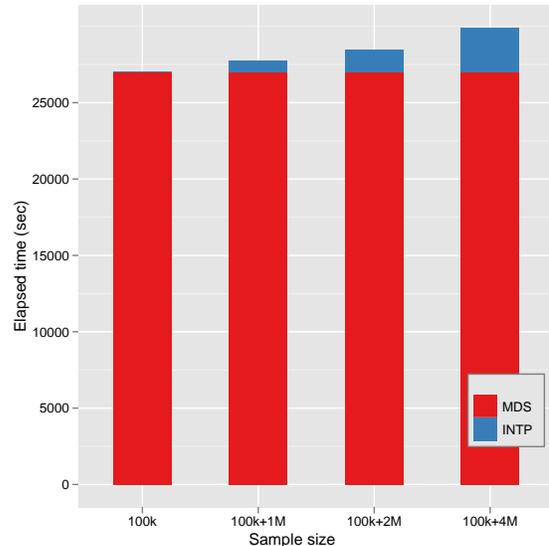


Figure 6. Running time of Out-of-Sample approach which combine full MDS running time with sample data ($M = 100k$) and MI-MDS running time with different out-of-sample data size, i.e. 1M, 2M, and 4M.

We also measure the runtime of MI-MDS algorithm with large-scale data set up to 4 million points. Fig. 6 shows the running time of out-of-sample approach in commulated bar graph, which represent full MDS running time of sample data ($M = 100k$) in red bar and MI-MDS interpolation time of out-of-sample data ($n = 1M, 2M, \text{ and } 4M$) in blue bar on top of the red bar. As we expected, the running time of MI-MDS is much faster than full MDS running time in Fig. 6. Although MI-MDS interpolation running time in Table I is much smaller than full MDS running time (27006 seconds), MI-MDS deals with much larger amount of points, i.e. 10, 20, and 40 times larger number of points. Note that we cannot run parallel SMACOF algorithm [17] with even 200,000 points on the tested system. Even though we assume that we are able to run parallel SMACOF algorithm with millions of points on the tested cluster system, the parallel SMACOF will take 100, 400, and 1600 times longer with 1M, 2M, and 4M data than the running time of parallel SMACOF with 100k data, due to the $\mathcal{O}(N^2)$ computational complexity. As opposed to the approximated full MDS running time, the proposed MI-MDS interpolation takes much less time to deal with millions of points than parallel SMACOF algorithm. In numeric, MI-MDS interpolation is faster than approximated full parallel MDS running time in 3693.5, 7454.2, and 14923.8 times with 1M, 2M, and 4M data, correspondingly.

If we extract the MI-MDS running time only with respect to the out-of-sample data size from Fig. 6, the running time should be proportional to the number of out-of-sample data since the sample data size is fixed. Table I shows the exact running time of MI-MDS interpolation method with respect

to the number of out-of-sample data size (n) based on the same sample data ($M = 100k$), and the running time is almost exactly proportional to the out-of-sample data size (n) as it should be.

V. CONCLUSION

Large-scale data analysis is a prominent research area due to the data explosion of almost every domains. Huge amounts of data are generated not only from the scientific and technical area but also from personal life activities, such as digital pictures, video clips, postings on a personal blog system or social network media, and so on. The dimension reduction algorithms aim to generate low-dimensional human-perceivable configuration which is very useful to investigate the high-dimensional data sets. Among many dimension reduction algorithms, we focus on multidimensional scaling (MDS) algorithm in this paper due to its robustness and high applicability.

We have worked on several ways to improve a well-known MDS algorithm, called SMACOF [15], [16], with respect to computing capability. For increasing the possible number of points generated new configuration in a target dimension, we have worked on parallelization of SMACOF algorithm. The parallelization enables SMACOF algorithm to deal with hundreds of thousands of points via distributed multicore cluster systems, such as 32 nodes with 768 cores. Although the parallel SMACOF implementation provides much more computing power, it cannot be affordable to configure millions of points since the computational complexity and memory requirement of SMACOF algorithm is still $\mathcal{O}(N^2)$. The proposed majorizing interpolation MDS (MI-MDS) make possible to generate a mapping of millions of points with a trade-off between the computing capacity and the mapping quality.

ACKNOWLEDGMENT

This work is partially funded by National Institutes of Health grant 1RC2HG005806-01 and Microsoft Research. We would like to thank Prof. Wild and Dr. Zhu for providing chemical compounds data.

REFERENCES

- [1] G. Fox, S. Bae, J. Ekanayake, X. Qiu, and H. Yuan, "Parallel data mining from multicore to cloudy grids," in *Proceedings of HPC 2008 High Performance Computing and Grids workshop*, Cetraro, Italy, July 2008.
- [2] J. B. Kruskal and M. Wish, *Multidimensional Scaling*. Beverly Hills, CA, U.S.A.: Sage Publications Inc., 1978.
- [3] I. Borg and P. J. Groenen, *Modern Multidimensional Scaling: Theory and Applications*. New York, NY, U.S.A.: Springer, 2005.
- [4] C. Bishop, M. Svensén, and C. Williams, "GTM: A principled alternative to the self-organizing map," *Advances in neural information processing systems*, pp. 354–360, 1997.
- [5] T. Kohonen, "The self-organizing map," *Neurocomputing*, vol. 21, no. 1-3, pp. 1–6, 1998.
- [6] E. Lessa, "Multidimensional analysis of geographic genetic structure," *Systematic Biology*, vol. 39, no. 3, pp. 242–252, 1990.
- [7] J. Tzeng, H. Lu, and W. Li, "Multidimensional scaling for large genomic data sets," *BMC bioinformatics*, vol. 9, no. 1, p. 179, 2008.
- [8] P. Groenen and P. Franses, "Visualizing time-varying correlations across stock markets," *Journal of Empirical Finance*, vol. 7, no. 2, pp. 155–172, 2000.
- [9] D. Agrafiotis, D. Rassokhin, and V. Lobanov, "Multidimensional scaling and visualization of large molecular similarity tables," *Journal of Computational Chemistry*, vol. 22, no. 5, pp. 488–500, 2001.
- [10] H. Lahdesmaki, X. Hao, B. Sun, L. Hu, O. Yli-Harja, I. Shmulevich, and W. Zhang, "Distinguishing key biological pathways between primary breast cancers and their lymph node metastases by gene function-based clustering analysis," *International journal of oncology*, vol. 24, no. 6, pp. 1589–1596, 2004.
- [11] W. S. Torgerson, "Multidimensional scaling: I. theory and method," *Psychometrika*, vol. 17, no. 4, pp. 401–419, 1952.
- [12] J. B. Kruskal, "Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis," *Psychometrika*, vol. 29, no. 1, pp. 1–27, 1964.
- [13] Y. Takane, F. W. Young, and J. de Leeuw, "Nonmetric individual differences multidimensional scaling: an alternating least squares method with optimal scaling features," *Psychometrika*, vol. 42, no. 1, pp. 7–67, 1977.
- [14] T. Hofmann and J. M. Buhmann, "Pairwise data clustering by deterministic annealing," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, pp. 1–14, 1997.
- [15] J. de Leeuw, "Applications of convex analysis to multidimensional scaling," *Recent Developments in Statistics*, pp. 133–146, 1977.
- [16] —, "Convergence of the majorization method for multidimensional scaling," *Journal of Classification*, vol. 5, no. 2, pp. 163–180, 1988.
- [17] J. Y. Choi, S.-H. Bae, X. Qiu, and G. Fox, "High performance dimension reduction and visualization for large high-dimensional data analysis," in *Proceedings of the 10th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID) 2010*, May 2010.
- [18] T. M. Cover and P. E. Hart, "Nearest neighbor pattern classification," *IEEE Transaction on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.
- [19] S.-H. Bae, J. Y. Choi, X. Qiu, and G. Fox, "Dimension reduction and visualization of large high-dimensional data via interpolation," in *Proceedings of the ACM International Symposium on High Performance Distributed Computing (HPDC) 2010*, Chicago, Illinois, June 2010.