



TECHNICAL NOTE

D-EE: Distributed software for visualizing intrinsic structure of large-scale single-cell data

Shaokun An ^{1,2}, Jizu Huang ^{1,2,*} and Lin Wan ^{1,2,*}

¹NCMIS, LSEC, LSC, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Zhongguancun East Road, Haidian District, Beijing, 100190, China and ²School of Mathematical Sciences, University of Chinese Academy of Sciences, Yuquan Road, Shijingshan District, Beijing, 100049, China

*Correspondence address. Jizu Huang, NCMIS, LSEC, LSC, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Zhongguancun East Road, Haidian District, Beijing, 100190, China. E-mail: huangjz@lsec.cc.ac.cn  <http://orcid.org/0000-0002-2324-0010> and Lin Wan, NCMIS, LSEC, LSC, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Zhongguancun East Road, Haidian District, Beijing, 100190, China. E-mail: lwana@amss.ac.cn  <http://orcid.org/0000-0002-3511-0512>

Abstract

Background: Dimensionality reduction and visualization play vital roles in single-cell RNA sequencing (scRNA-seq) data analysis. While they have been extensively studied, state-of-the-art dimensionality reduction algorithms are often unable to preserve the global structures underlying data. Elastic embedding (EE), a nonlinear dimensionality reduction method, has shown promise in revealing low-dimensional intrinsic local and global data structure. However, the current implementation of the EE algorithm lacks scalability to large-scale scRNA-seq data. **Results:** We present a distributed optimization implementation of the EE algorithm, termed distributed elastic embedding (D-EE). D-EE reveals the low-dimensional intrinsic structures of data with accuracy equal to that of elastic embedding, and it is scalable to large-scale scRNA-seq data. It leverages distributed storage and distributed computation, achieving memory efficiency and high-performance computing simultaneously. In addition, an extended version of D-EE, termed distributed optimization implementation of time-series elastic embedding (D-TSEE), enables the user to visualize large-scale time-series scRNA-seq data by incorporating experimentally temporal information. Results with large-scale scRNA-seq data indicate that D-TSEE can uncover oscillatory gene expression patterns by using experimentally temporal information. **Conclusions:** D-EE is a distributed dimensionality reduction and visualization tool. Its distributed storage and distributed computation technique allow us to efficiently analyze large-scale single-cell data at the cost of constant time speedup. The source code for D-EE algorithm based on C and MPI tailored to a high-performance computing cluster is available at <https://github.com/ShaoKunAn/D-EE>.

Keywords: dimensionality reduction; distributed storage; distributed computation; large-scale data; single-cell sequencing

Background

The advent of single-cell sequencing provides high-dimensional profiles of cellular states at single-cell resolutions (e.g., single-cell RNA sequencing [scRNA-seq] of transcriptomes), offering the opportunity to unveil intrinsic biological processes and mechanisms. Dimensionality reduction and visualization methods have been extensively studied because they play vital roles in revealing the intrinsic structures underlying scRNA-seq high-

dimensional data [1]. Nonetheless, it is still challenging for these state-of-the-art methods of dimensionality reduction and visualization to preserve both local and global structures of data in low-dimensional space. For example, the celebrated t-distributed stochastic neighbor embedding (t-SNE) algorithm [2] is widely used in the single-cell community [3]. It emphasizes the preservation of local structures, but it often distorts global structures [4–6]. As a solution, the Uniform Manifold Approxi-

Received: 9 August 2020; Revised: 29 September 2020; Accepted: 20 October 2020

© The Author(s) 2020. Published by Oxford University Press GigaScience. This is an Open Access article distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted reuse, distribution, and reproduction in any medium, provided the original work is properly cited.

mation and Projection (UMAP) algorithm [7] was developed, with the aim to preserve global structures, drawing increasing attention in the single-cell data analysis community [8]. However, a recent study showed that UMAP does not improve upon t-SNE in this regard when using the same initialization [9], making the validity of UMAP debatable.

In contrast, elastic embedding (EE), a nonlinear dimensionality reduction method, attempts to preserve both local and global structures underlying the data [4]. To achieve this goal, EE penalizes the placement of latent points in close proximity away from dissimilar data points in high-dimensional space, thus resolving the difficulty of global structure preservation (see [4], or Methods for details). EE has attracted increasing interest among statistical researchers [10]. It has also shown remarkable performance on visualizing the intrinsic structures of scRNA-seq data [1, 5, 11]. However, the current implementations of the EE algorithm are not scalable to sample size N (e.g., number of cells). Thus, it cannot be used for large-scale scRNA-seq datasets. For example, the storage of the attractive and the repulsive weight matrices of the EE algorithm is $\mathcal{O}(N^2)$.

Therefore, we present a distributed optimization implementation of EE, termed D-EE. D-EE not only reveals the low-dimensional intrinsic structures of data with the same accuracy as EE but also is scalable to large-scale scRNA-seq data. It leverages distributed storage and distributed computation, achieving memory efficiency and high-performance computing simultaneously (Fig. 1). In addition, a distributed optimization implementation of the time-series EE (TSEE) algorithm [11], termed D-TSEE, is also provided for visualizing large-scale time-series scRNA-seq data. In this study, we demonstrate the power of D-EE and D-TSEE on both simulated and real datasets. Both D-EE and D-TSEE (1) achieve the same accuracy as EE and TSEE, respectively; and (2) gain high strong scaling performance on large-scale datasets.

Methods

Elastic embedding algorithm

EE was proposed by Carreira-Perpiñán [4]. It optimizes an energy function containing the attractive and repulsive terms.

Given N samples $\mathbf{Y} = \{y_1, y_2, \dots, y_N\}$, where $y_i \in \mathbb{R}^D$ represents its high-dimensional coordinates, the goal of EE is to map the data from high-dimensional space onto a low-dimensional representation $\mathbf{X} = \{x_1, x_2, \dots, x_N\}$ with $x_i \in \mathbb{R}^d$ and $d \ll D$ by minimizing an energy function

$$E(\mathbf{X}, \lambda) = \sum_{m,n=1}^N w_{nm}^+ \|x_n - x_m\|^2 + \lambda \sum_{m,n=1}^N w_{nm}^- \exp(-\|x_n - x_m\|^2),$$

where $w_{nm}^+ = \exp[-(1/2)\|y_n - y_m\|^2/\sigma_n^2]$ and $w_{nm}^- = \|y_n - y_m\|^2$. The first term acts as an attractive force to preserve local distances, while the second term acts as a repulsive force to preserve global structures or to separate latent points. The parameter $\lambda \in \mathbb{R}^+$ trades off the 2 terms, and a larger value implies that preservation of global structures is more important. In single-cell data analysis, with $\lambda = 10$, EE can achieve robust performance with high accuracy [5]. Therefore, we set the default value of λ as 10 for D-EE.

The σ_m in w_{nm}^+ is a sample-specific scaling parameter. It is estimated adaptively by solving a sample-specific root-finding problem, such that the sample-specific distribution over its neighbors has a desired perplexity (see [12] for details). We set a default value of perplexity as 20 in this study. It is worth not-

ing that a newly proposed combinational perplexity has been applied to t-SNE [3], which greatly enhances the performance of t-SNE in preservation of global structures. The combinational perplexity can be also adopted by D-EE in a future update.

An extension of EE, TSEE [11], was recently proposed to handle the dimensionality reduction problems of time-series scRNA-seq data. It works by minimizing

$$E(\mathbf{X}, \lambda) = \sum_{m,n=1}^N w_{nm}^+ \|x_n - x_m\|^2 + \lambda \sum_{m,n=1}^N (w_{nm}^- + \beta t_{nm}) \exp(-\|x_n - x_m\|^2),$$

where t_{nm} represents the dissimilarity of time of pairwise points, and β trades off the weights between dissimilarities of time stages and expression space.

Numerical optimization of EE

Because the optimization solution of TSEE is basically the same as that of EE, we only give the numerical solution of EE. First, we denote $\mathbf{W}_P = \{w_{nm}^+\}$ and $\mathbf{W}_N = \{w_{nm}^-\}$. Owing to the existence of parameters $\{\sigma_n\}$, \mathbf{W}_P is not a symmetrical matrix but we make it symmetric by taking $\mathbf{W}_P := \mathbf{W}_P + \mathbf{W}_P^T$. Next, the diagonal elements of \mathbf{W}_P and \mathbf{W}_N are set to zero. Finally, each element is normalized by being divided by the sum of all elements in the matrix.

To solve the optimization problem, the classic quasi-Newton methods update \mathbf{X}_{k+1} according to $\mathbf{X}_{k+1} = \mathbf{X}_k + \alpha_k \mathbf{P}_k$ in the k th iteration, where α_k is the step length determined by a line search procedure and \mathbf{P}_k is the search direction obtained by solving the Jacobian system $\mathbf{B}_k \mathbf{P}_k = -\mathbf{G}_k$. In this equation, \mathbf{B}_k is positive-definite to guarantee the decrease of the objective function. $\mathbf{G}_k = \mathbf{L}_k \mathbf{X}_k$ is the gradient of the objective function in the k th iteration, where \mathbf{L}_k is the Laplacian of $\mathbf{W}_k = \{w_{nm}^{(k)}\}$ with $w_{nm}^{(k)} = w_{nm}^+ - \lambda w_{nm}^- \exp(-\|x_n^{(k)} - x_m^{(k)}\|^2)$. These procedures are repeated until a certain termination criterion is satisfied. During the iteration, \mathbf{B}_k generally needs to be updated in each iteration as well.

To solve the EE-like optimization problems, a technique termed "partial-Hessian optimization strategies" has been proposed to use partial information of the Hessian \mathbf{L}_P [13], which is the Laplacian of \mathbf{W}_P and is invariant in each iteration. This invariance makes it possible to utilize some precondition approaches, e.g., lower-upper (LU) decomposition, to improve calculation efficiency. The effectiveness of the determined direction, called "spectral direction", has been validated experimentally in previous work [13].

D-EE algorithm

We provide a distributed optimization implementation of EE, termed D-EE. The overview of the newly proposed D-EE algorithm is given in Fig. 1. During whole optimization implementation, multiple processes are used for computation and storage of data. In Fig. 1, 2 processes, \mathcal{P}_0 and \mathcal{P}_1 , are taken as an example. To achieve high performance in computing and memory efficiency simultaneously, our proposed distributed algorithm divides data (\mathbf{W}_P , \mathbf{W}_N , and \mathbf{G}_k) by rows for the multiple processes assigned. To avoid frequent communication, the whole original high-dimensional dataset \mathbf{Y} is read and stored in each process, and the low-dimensional embedding \mathbf{X} is established in each process as well because the storage consumed by \mathbf{Y} and \mathbf{X} is much less when compared to other $N \times N$ matrices used during computation. It is worth noting that, because most of the com-

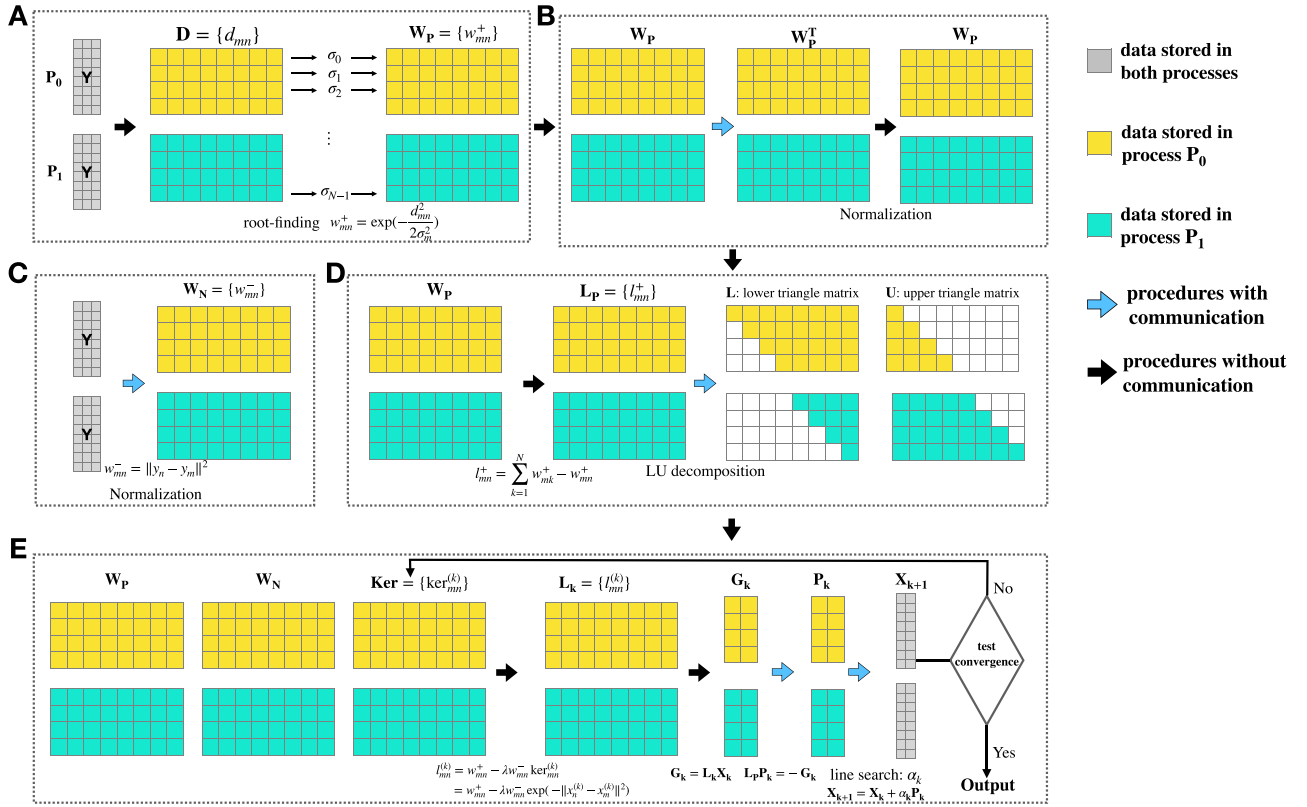


Figure 1 Overview of D-EE algorithm. The D-EE algorithm can be decomposed into 5 parts. Part A: the computation of matrices D and W_P . To obtain W_P , the parameters $\{\sigma_n\}$ are determined by solving a series of root-finding problems. Part B: the symmetry and normalization of the matrix W_P . Part C: the computation and normalization of W_N . Part D: the computation of the Laplacian matrix L_P together with LU decomposition. Part E: the computation of X_{k+1} by solving an optimization problem with the classic Quasi-Newton methods iteratively. In the k th iteration, L_k is computed on the basis of W_P , W_N , and X_k , which are used to obtain the gradient $G_k = L_k X_k$. The descent direction P_k is then determined by solving a linear system $L_P P_k = -G_k$. Finally, X_{k+1} is updated according to $X_{k+1} = X_k + \alpha_k P_k$, where the step size α_k is calculated by a line search method.

putation of each row in 1 matrix generally merely depends on the same row of other matrices (see the approximated computational complexity of D-EE in the following section for details), the partition procedure that we design in the D-EE algorithm is an almost optimal partition in parallel computing as a result of the optimal leverage of computation and communication. On the one hand, the total computational cost of the D-EE algorithm is almost the same as that of the centralized algorithm of EE. On the other hand, most procedures in the D-EE algorithm are communication-free, as indicated in Fig. 1 by the black arrows. Even though some procedures still exist with communication, as shown in Fig. 1 by blue arrows, the communication volume is on a much lower order than the cost of computation.

Computation of matrices D , W_P , W_N

As mentioned before, the matrices W_P , W_N depend on the high-dimensional data Y and $\{\sigma_n\}_{n=1}^N$. Because each σ_n is obtained by solving a root-finding problem from the n th row of the distance matrix D , each matrix is equally, or almost equally, partitioned into multiple nonoverlapping parts by rows and stored in multiple processes, as shown in Fig. 1A. Let us denote $D = [D^1, \dots, D^P]$, where submatrix D^i with size of $M_i \times N$ is stored in the i th process and P is the number of processes we used. The $[\dots]$ represents a column vector. Similar notations are used for the other $N \times N$ matrices. It is clear that each row of matrices D , W_P , W_N depends on all original high-dimensional data Y .

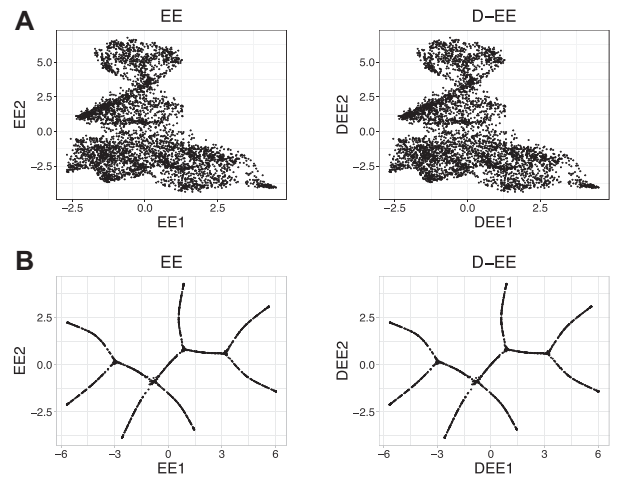


Figure 2 D-EE and EE achieve the same results on 2 datasets when using the same initializations. A: The 2-D mapping of HSPC data obtained by the 2 algorithms. B: The 2-D mapping of PHATE data obtained by the 2 algorithms.

Therefore, we load a copy of Y into each process to avoid frequent communication.

In the centralized implementation of EE, the parameters σ_n , $n = 1, \dots, N$, are calculated by iteratively solving a sequence of root-finding problems. The iteration method for the root-finding

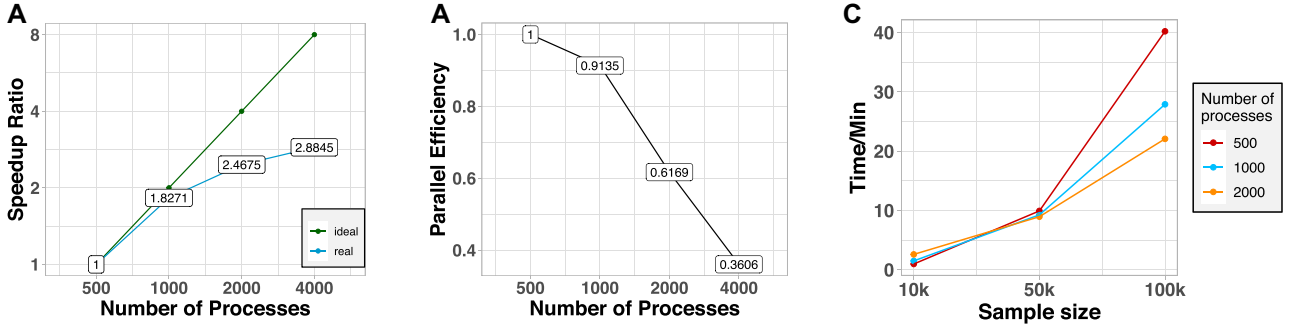


Figure 3 Strong scaling results and parallel efficiency of the D-EE algorithm on an LSSC-IV supercomputer. We apply D-EE on the iPSC dataset by using 500, 1,000, 2,000, and 4,000 processes, respectively. A: The strong speedup ratio increases with increase in the number of processes. The green line represents the ideal speedup ratio and the blue line represents the speedup ratio obtained by D-EE. B: The parallel efficiency decreases at an acceptable rate with the increase in number of processes. C: Computational times consumed for 10,000, 50,000, 100,000 samples under 500, 1,000 and 2,000 processes, respectively.

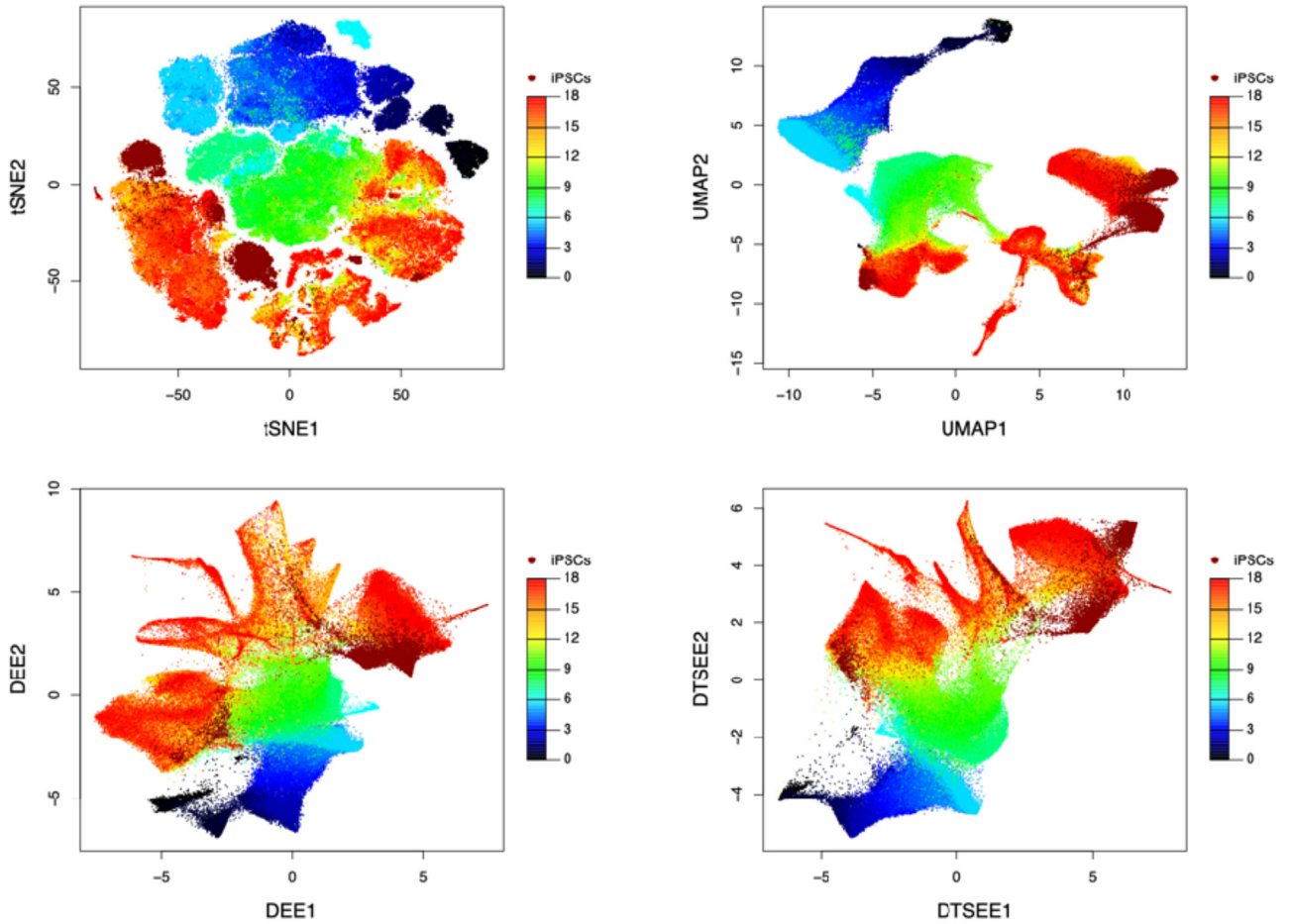


Figure 4 Cells are colored by time stages in the iPSC dataset on the 2D space obtained by 4 dimensionality reduction methods, i.e., t-SNE, UMAP, D-EE, and D-TSEE.

problems is improved by reordering the computation of $\{\sigma_n\}_{n=1}^N$ according to the distances of all samples (Y), which is also the complete distance matrix D [12]. Then the reordered root-finding problems are sequentially solved by taking the solution of the previous one as the initial value of the next. Because the parameters are distributed in different processes, it is clear that the sequential root-finding approach cannot be parallelized without modifications. In the D-EE algorithm, we calculate $\{\sigma_n\}_{n=1}^N$ in the following parallel way. First, we decompose $\{\sigma_n\}_{n=1}^N$ into P subsets

as $\Sigma_i = \{\sigma_n\}_{n=\mathcal{M}_i+1}^{\mathcal{M}_{i+1}}$ with $i = 0, \dots, P - 1$. The elements in the i th subset Σ_i are computed and stored in the i th process. Similar to the centralized algorithm of EE, we then reorder Σ_i according to the distance matrix D^i and iteratively solve the corresponding root-finding problems within the i th process. According to the distributions of the initial data Y and the matrices established before, we conclude that the D-EE algorithm calculates $\{\sigma_n\}_{n=1}^N$ in parallel, which is communication-free. The efficiency of the root-finding approach is also guaranteed by the local order. Be-

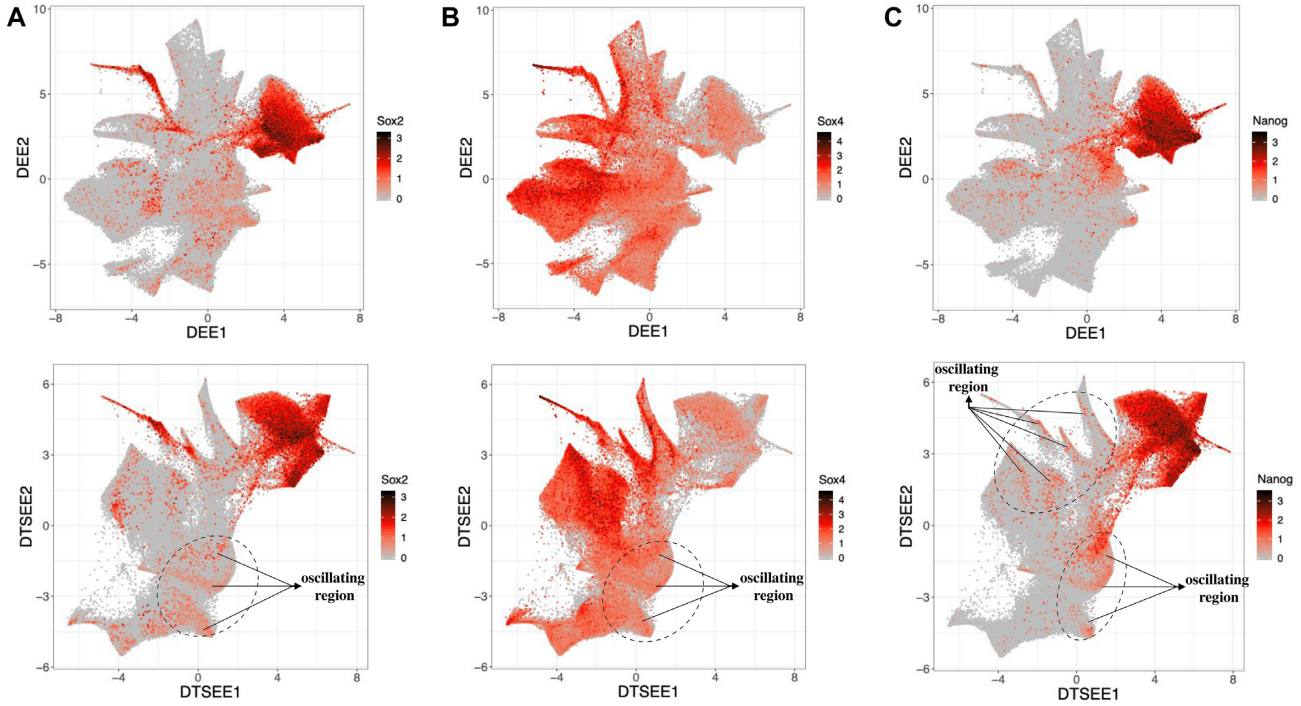


Figure 5 Cells are colored by gene expression of Sox2, Sox4, and Nanog in the iPSC dataset on the 2D embeddings obtained by D-EE and D-TSEE, respectively.

cause we only change the order and initial guesses of the root-finding problems, the solutions of the root-finding problems, as obtained from D-EE, are almost the same as those from EE. With the whole original high-dimensional dataset Y and the subset Σ_i , we can compute the following submatrices W_P^i , W_N^i in the i th process. Thus, we give a parallel and communication-free approach to compute matrices D , W_P , W_N .

Normalization of W_P and W_N

After computing matrices W_P , W_N , each process sets the diagonal elements belonging to it as 0 in parallel. Then, we set $W_P := W_P + W_P^T$ such that W_P becomes a symmetric matrix. Let us denote $W_P^T := \hat{W}_P = [\hat{W}_P^1, \dots, \hat{W}_P^P]$, where submatrix \hat{W}_P^i has the size of $M_i \times N$. In the i th process, we first obtain the elements of the submatrix \hat{W}_P^i from the other $P - 1$ processes by communication and then compute $W_P^i := W_P^i + \hat{W}_P^i$. Here point-to-point communication happens, and the communication volume for each process is $\mathcal{O}(N^2/P)$.

To normalize the matrices W_P , W_N , each element should be divided by the sum of all elements in the matrix. The sum of all elements in matrix W_P is parallel computed as follows. First, each process calculates the sum of all elements in the submatrix W_P^i independently. We denote the sum of all elements in the submatrix W_P and W_P^i as S and S^i , respectively. Then we compute the sum of all elements in matrix W_P by $S = \sum_{i=1}^P S_i$ through an `MPLAllgather` action. Here all-to-all communication happens, and the communication volume for each process is $\mathcal{O}(P)$. Then, we normalize matrix W_P in each process by taking $W_P^i = W_P^i/S$ in parallel without communication. The normalization of matrix W_N is done in a similar way.

Computation of low-dimensional embedding X

After normalizing W_P , its Laplacian L_P , which is needed for the subsequent determination of descent direction, is computed in parallel as follows. In the i th process, we calculate the elements of submatrix L_P^i by using $l_{mn}^+ = \sum_{k=1}^N w_{mk}^+ - w_{mn}^+$, where l_{mn}^+ and w_{mn}^+ are the elements of matrices L_P and W_P , respectively. Because the 2 matrices are partitioned by row in the same way, the computation of L_P is also communication-free.

The low-dimensional embedding X is obtained by solving the optimization problem with the partial-Hessian optimization strategy. During the quasi-Newton procedures, the dense linear system $L_P P_k = -G_k$ must be solved in parallel. In the D-EE algorithm, we perform LU decomposition on L_P . Considering that L_P is positive semi-definite but not positive definite, a small value μ is added to the diagonal of L_P in practice. During the following sections, we still use L_P to denote the adjusted matrix. LU decomposition on $L_P = LU$ is done with PETSc, which provides uniform and efficient access to all linear system solvers in the package, including parallel and sequential, direct, and iterative [14–16]. Here, L and U are the corresponding lower and upper triangle matrices, respectively. With the decomposition of LU, the dense linear system $L_P P_k = -G_k$ is replaced by 2 sublinear systems $L \hat{P}_k = -G_k$ and $U P_k = \hat{P}_k$, which can be solved by the backward substitution method.

As shown in Fig. 1D, the partitions of L_P , L , and U are the same as W_P . Let us denote $P_k = [P_k^1, \dots, P_k^P]$, where submatrix P_k^i with size of $M_i \times d$ is stored in the i th process, and a similar partition is performed on G_k . Based on the partitions of L_P , L , U , P_k , and G_k , the computational complexities per process of LU decomposition and backward substitution are $\mathcal{O}(N^3/P)$ and $\mathcal{O}(N^2/P)$, with corresponding communication volumes of

$\mathcal{O}(N^2/P)$ and $\mathcal{O}(N/P)$, respectively. According to the analysis, LU decomposition is only done in the first iteration of the quasi-Newton method, and matrices \mathcal{L} and \mathcal{U} are stored and reused during the whole quasi-Newton procedure.

The gradient \mathbf{G}_k on the right-hand side of the linear system $\mathbf{L}_p \mathbf{P}_k = -\mathbf{G}_k$ is calculated by $\mathbf{G}_k = \mathbf{L}_k \mathbf{X}_k$, where the $N \times N$ matrix \mathbf{L}_k depends on matrices \mathbf{W}_p , \mathbf{W}_N , and \mathbf{Ker} . Here the elements of matrix \mathbf{Ker} are defined as $\ker_{mn} = \exp(-\|x_m - x_n\|^2)$, and the elements of matrix \mathbf{L}_k are defined as $l_{mn}^{(k)} = w_{mn}^+ - \lambda w_{mn}^- \ker_{mn}^{(k)}$. As shown in Fig. 1E, the partitions of matrices \mathbf{L}_k and \mathbf{Ker} are the same as those of \mathbf{W}_p . In D-EE, we store all elements of \mathbf{X}_k in each process, which is the same as the original high-dimensional data \mathbf{Y} . Thus, we can parallel-compute matrices \mathbf{L}_k and \mathbf{Ker} in the same way with the matrix \mathbf{W}_N , which means the procedure is also communication-free.

After solving the linear system, we obtain the search direction \mathbf{P}_k . Then, we update \mathbf{X}_{k+1} according to $\mathbf{X}_{k+1} = \mathbf{X}_k + \alpha_k \mathbf{P}_k$, where α_k is determined by a line search approach. As mentioned before, the low-dimensional embedding \mathbf{X}_k is stored sequentially, but \mathbf{P}_k is distributed stored. Thus, we first compute the elements of submatrix \mathbf{P}_{k+1}^i in the i th process and then gather all elements of \mathbf{P}_k in each process by the all-gather function in MPI. Here all-to-all communication happens, and the order of communication volume for each process is $\mathcal{O}(Nd)$. In line search steps, we need to calculate the energy function $E(\mathbf{X}, \lambda)$ several times, which is computed in parallel according to the following formula:

$$E(\mathbf{X}, \lambda) = \sum_{i=1}^P \left\{ \sum_{m=\mathcal{M}_i+1}^{\mathcal{M}_{i+1}} \sum_{n=1}^N \left[w_{mn}^+ \|x_m - x_n\|^2 + \lambda w_{mn}^- \exp(-\|x_m - x_n\|^2) \right] \right\}.$$

The summation included in the curly braces is calculated in each process simultaneously and then gathered by the MPI all-gather function. Here all-to-all communication happens, and the communication volume for each process is $\mathcal{O}(P)$.

Results

Data description

We test the accuracy and scalability of D-EE on 3 datasets. The first simulated dataset [17], named PHATE data for convenience, consists of 1,440 samples and 60 features. It is a complex tree structure that simulates a cellular developmental process, namely, progressions, branch or split in progressions, and end state of progression, composed of 10 branches in total. We first perform principal component analysis (PCA) on the original data, reserving a 1,440 samples \times 7 features matrix.

The second dataset characterizes the process of mouse hematopoietic stem and progenitor cells (HSPC) bifurcating to myeloid and erythroid precursors [18], consisting of 4,423 samples. The obtained single-cell read count dataset is pre-processed by the Seurat package (version 3.2.1) [19, 20]. First, we normalize the gene expression in each sample as follows: we divide each gene read count by the total read counts for each cell and then multiply by a scale factor of 10^4 and add 1, followed by taking a logarithmic transformation. Second, we select the top 2,000 variable genes using the default “vst” method of the Seurat package, i.e., variance-stabilizing transformation [21]. Finally, we conduct PCA on the processed data and select the top

50 largest principal components, resulting in a 4,423 samples \times 50 features matrix as input of EE and D-EE.

The third dataset is a large-scale time-series scRNA-seq dataset containing $\sim 250,000$ cells [22]. The data characterize reprogramming of fibroblasts to induced pluripotent stem cells (iPSC), which were collected at half-day intervals across 18 days, resulting in 39 time points. Because the final time point of the iPSC status was not annotated temporally, we therefore set the final point as 20th day for the input to D-TSEE. We pre-process these data with the Seurat package as well. Same as the pre-processing of the HSPC data, we first filter cells and genes to include cells where ≥ 200 features are detected and to include genes detected in ≥ 50 cells, obtaining 259,081 cells and 19,427 genes. After that, we perform logarithmic transformation, select variable features, and perform PCA as described in the HSPC dataset, obtaining a 259,081 samples \times 50 features matrix as input for the dimensionality reduction methods.

D-EE achieves high strong scaling efficiency

We evaluate D-EE using both simulated and real scRNA-seq datasets. The numerical tests are carried out on the LSSC-IV supercomputer. The 400 computing nodes of LSSC-IV comprise 2 18-core Intel Xeon Gold CPUs with 192 GB local memory and are interconnected via a proprietary high-performance network. First, we use PHATE data and HSPC data to test the consistency between D-EE and EE results. We use 36 processes in both D-EE algorithms. The low dimensions are set to 2 for the convenience of visualization for both datasets, and the parameter λ used is set to the default value 10. We use the same initialization generated from the Gaussian distribution as that in the original EE Matlab code.

D-EE achieves 2D embedding consistent with that of EE (Fig. 2). To measure the consistency quantitatively, we calculate the relative error, which is defined by

$$\text{relative error} = \frac{\|A - B\|_F}{\|A\|_F},$$

where $\|\cdot\|_F$ is the Frobenius norm of the matrix and A and B represent the output of EE and D-EE, respectively. The Frobenius norm of a matrix $A \in \mathbb{R}^{m \times n}$ is defined as

$$\|A\|_F = \sqrt{\sum_{i=1}^m \sum_{j=1}^n |a_{ij}|^2}.$$

Their relative errors of D-EE in the HSPC dataset and PHATE dataset are 2.42×10^{-6} and 1.60×10^{-6} , respectively, thus further validating the consistency of results by D-EE and EE.

To test the performance of parallel efficiency of D-EE on the large-scale dataset, we apply D-EE to the iPSC dataset ($\sim 250,000$ cells) using 500, 1,000, 2,000, and 4,000 processes, respectively, and for each setting of number of processes we run it at least twice. The averaged computation times of D-EE for the iPSC dataset are 5.83, 3.19, 2.36, and 2.02 hours, when the numbers of processes are 500, 1,000, 2,000, and 4,000, respectively. Next, we evaluate the parallel performance of D-EE based on 2 widely used indexes, the strong scaling speedup ratio and the parallel efficiency. The strong speedup ratio is defined as

$$S = \frac{T_s}{T_p},$$

where T_s and T_p are the time of computation by using a single process and p processes, respectively. The parallel efficiency is defined as

$$E = \frac{S}{p} = \frac{T_s}{pT_p}.$$

The ideal strong speedup should be p , and the corresponding parallel efficiency should be 1 when p processes are used. However, it is impossible to run the data with 250,000 samples on a single process owing to limited memory and the low efficiency of the EE algorithm. Thus, we take the time of computation of 500 processes as T_s with $s = 500$, and the parallel efficiency is then reformulated as

$$E = \frac{sT_s}{pT_p}.$$

When adopting the speedup ratio and parallel efficiency as the indexes for scaling, a strong scaling performance at remarkable speedup is observed when increasing CPUs from 500 to 4,000 processes (Fig. 3A and B).

We further evaluate the performance of D-EE on computational times in our supercomputer. Three test cases with sample sizes of 10,000, 50,000, and 100,000 are run on 500, 1,000, and 2,000 processors, respectively. It is shown that when the number of processes is held constant, it is natural that the computational time increases as total sample size increases (as shown in Fig. 3C).

In practice, for analysis of a large-scale single-cell dataset, a workstation with multiple CPUs and large memory is recommended. Meanwhile, we also test D-EE on datasets at different sample sizes and find that D-EE can be efficiently implemented on a typical personal computer (e.g., 8 CPU threads, 16 GB RAM) with a sample size of $\leq 12,000$, while on a conventional workstation (e.g., 40 CPU threads, 256 GB RAM) with a sample size of $\leq 48,000$.

D-EE and D-TSEE recover intrinsic low-dimensional structures of large-scale scRNA-seq data

We illustrate the application of D-EE and D-TSEE on the large-scale iPSC dataset. We visualize the iPSC data on the 2D space using t-SNE, UMAP, D-EE, and D-TSEE, respectively (Fig. 4). The same pre-processed single-cell data are used for all 4 methods. The t-SNE is conducted by the Fit-SNE method [23] with a PCA initialization, and we choose the learning rate as 1/12 of the sample size (according to [3]) for better preservation of the global structures. UMAP is conducted by adjusting the number of nearest neighbors (NNs) to balance the preservation of local and global structures. We find that UMAP is not sensitive when choosing the number of NNs from 30 (defaults) to 500 (square root of the number of samples), and we thus set the number of NNs to be 100 in our study. Both t-SNE and UMAP are implemented by the Seurat software (version 3.2.1). In our implementations, both D-EE and D-TSEE are used with their default parameters.

We color the cells on the basis of time stages on their low-dimensional embeddings obtained by the 4 dimensionality reduction results (Fig. 4). We find that t-SNE preserves the time lineage structure of data in an “S” shape with small gaps (Fig. 4, upper left panel); UMAP also shows a time lineage structure but with a large gap occurring between time stages 5.5 and 6 (Fig. 4, upper right panel). In contrast, both D-EE and D-TSEE preserve

continuous time lineage structures in low-dimensional space (Fig. 4, lower panels).

We further explore the gene expression patterns of Sox2, Sox4, and Nanog on both D-EE and D-TSEE embeddings (Fig. 5). These genes are key regulators during stem cell differentiation and reprogramming process [24–26]. Previous study has shown evidence that these genes may oscillate during cell development progression [11, 26]. These genes display oscillatory gene expression patterns in the early stage of iPSC on the D-TSEE view (Fig. 5), providing useful information and clues for downstream analysis.

Conclusion

In this work, we develop a novel tool, D-EE, for visualizing large-scale scRNA-seq data. D-EE implements the distributed storage and distributed computing techniques to a powerful nonlinear dimensionality reduction method, EE. The optimal distributed computational strategies implemented by D-EE allow it to achieve not only strong scalability on large-scale datasets but also the exact optimization solution as original EE by fully utilizing the whole data. Numerical experiments validate the correctness and parallel efficiency of D-EE. Considering the emergence of time-series scRNA-seq data, our D-TSEE tool allows us efficiently to perform dimensionality reduction on large-scale single-cell data by using experimentally temporal information. Besides, when incorporating temporal information if it is available, D-TSEE can reveal dynamic gene expression patterns, providing insights for subsequent analysis of molecular mechanisms and dynamic transition progression.

We demonstrate that D-EE and D-TSEE work efficiently on large-scale datasets at a supercomputer. However, the proposed distributed algorithm D-EE still has disadvantages due to the huge computational cost and storage with a relatively large number of cells. Therefore, D-EE is limited to handling and analyzing huge-scale datasets with the number of cells up to the order of millions [27]. In comparison, the state-of-the-art accelerated implementations of t-SNE (e.g., Fit-SNE) and UMAP are of close-to-linear computational complexities, showing great efficiency in huge data analysis.

In a future study, to resolve the limitation of D-EE on huge-scale data computation, we can accelerate D-EE by adopting either the fast Fourier transform as used in Fit-SNE, or adopting the state-of-the-art neural network framework used by net-SNE [28]. On the other hand, because a huge-scale single-cell dataset can be highly redundant, we can also select a subset of informative samples using an advanced geometric sketching tool [29] prior to application of D-EE.

Availability of Source Code and Requirements

- Project name: D-EE
- Project home page: <https://github.com/ShaoKunAn/D-EE>
- Operating systems: Linux
- Programming language: C, R
- Other requirements: Multi-core processor, implementation of MPI library (i.e., OpenMPI or IntelMPI) installed on each node of the cluster, a reasonably fast interconnecting infrastructure, PETSc 3.11.4 or higher
- License: GNU General Public License
- biotools: d-ee
- RRID:SCR_019058

Availability of Supporting Data and Materials

The PHATE data supporting the results of this article are available in the GitHub repository [17]. The iPSC data are available in the NCBI repository with No. GSE122662 [22]. The HSPC data are available in the NCBI repository with accession No. GSE72857 and the dataset used in our study is downloaded from their GitHub repository [30]. An archival copy of the code is available via the GigaScience database, GigaDB [31].

Abbreviations

CAS: Chinese Academy of Sciences; CPU: central processing unit; D-EE: distributed optimization implementation of elastic embedding; D-TSEE: distributed optimization implementation of time-series elastic embedding; EE: elastic embedding; HSPC: hematopoietic stem and progenitor cells; LU: lower-upper; NCBI: National Center for Biotechnology Information; NNs: nearest neighbors; scRNA-seq: single-cell RNA sequencing; PCA: principal component analysis; PETSc: Portable, Extensible Toolkit for Scientific Computation; PHATE: Potential of Heat-diffusion for Affinity-based Transition Embedding; RAM: random access memory; TSEE: time-series elastic embedding; t-SNE: t-distributed stochastic neighbor embedding; UMAP: Uniform Manifold Approximation and Projection.

Competing Interests

The authors declare that they have no competing interests.

Funding

This work is supported by the National Key R&D Program of China under Grant 2018YFB0704304, NSFC grants (Nos. 11871069, 12071466), NCMIS of Chinese Academy of Sciences (CAS), LSEC of CAS, LSC of CAS, and the Youth Innovation Promotion Association of CAS.

Authors' Contributions

Conceptualization and Methodology: S.A., L.W. Software: S.A., J.H. Supervision: S.A., L.W., J.H. Funding Acquisition: L.W., J.H. Writing—Original Draft Preparation: S.A. Writing—Review & Editing: all authors.

References

- Hie B, Peters J, Nyquist SK, et al. Computational methods for single-cell RNA sequencing. *Annu Rev Biomed Data Sci* 2020;**3**(1):339–64.
- van der Maaten LJP, Hinton GE. Visualizing high-dimensional data Using t-SNE. *J Mach Learn Res* 2008;**9**: 2579–25.
- Kobak D, Berens P. The art of using t-SNE for single-cell transcriptomics. *Nat Commun* 2019;**10**:5416.
- Carreira-Perpiñán MÁ. The elastic embedding algorithm for dimensionality reduction. In: *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, Haifa, Israel. 2010:167–74.
- Chen Z, An S, Bai X, et al. DensityPath: an algorithm to visualize and reconstruct cell state-transition path on density landscape for single-cell RNA sequencing data. *Bioinformatics* 2019;**35**(15):2593–601.
- Nguyen LH, Holmes S. Ten quick tips for effective dimensionality reduction. *PLoS Comput Biol* 2019;**15**(6), doi:10.1371/journal.pcbi.1006907.
- McInnes L, Healy J, Melville J. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *arXiv* 2018:1802.03426.
- Becht E, McInnes L, Healy J, et al. Dimensionality reduction for visualizing single-cell data using UMAP. *Nat Biotechnol* 2019;**37**(1):38–44.
- Kobak D, Linderman GC. UMAP does not preserve global structure any better than t-SNE when using the same initialization. *bioRxiv* 2019, doi:10.1101/2019.12.19.877522.
- Wasserman L. Topological data analysis. *Annu Rev Stat Appl* 2018;**5**(1):501–32.
- An S, Ma L, Wan L. TSEE: an elastic embedding method to visualize the dynamic gene expression patterns of time series single-cell RNA sequencing data. *BMC Genomics* 2019;**20**(2):224.
- Vladymyrov M, Carreira-Perpiñán M. Entropic affinities: properties and efficient numerical computation. In: Dasgupta S, McAllester D, eds. *Proceedings of the 30th International Conference on Machine Learning, Atlanta, GA*. JMLR; 2013:477–85.
- Vladymyrov M, Carreira-Perpinan M. Partial-Hessian strategies for fast learning of nonlinear embeddings. *arXiv* 2012:1206.4646.
- Balay S, Abhyankar S, Adams MF, et al. PETSc. 2019. <https://www.mcs.anl.gov/petsc>. Accessed 15 July 2020.
- Balay S, Abhyankar S, Adams MF, et al. PETSc Users Manual. Argonne National Laboratory; 2020. <https://www.mcs.anl.gov/petsc/petsc-current/docs/manual.pdf>. Accessed 4 November 2020.
- Balay S, Gropp WD, McInnes LC, et al. Efficient management of parallelism in object oriented numerical software libraries. In: Arge E, Bruaset AM, Langtangen HP, eds. *Modern Software Tools in Scientific Computing*. Birkhäuser; 1997:163–202.
- Moon KR, van Dijk D, Wang Z, et al. Visualizing structure and transitions in high-dimensional biological data. *Nat Biotechnol* 2019;**37**(12):1482–92.
- Setty M, Tadmor MD, Reich-Zeliger S, et al. Wishbone identifies bifurcating developmental trajectories from single-cell data. *Nat Biotechnol* 2016;**34**(6):637–45.
- Butler A, Hoffman P, Smibert P, et al. Integrating single-cell transcriptomic data across different conditions, technologies, and species. *Nat Biotechnol* 2018;**36**(5): 411–20.
- Stuart T, Butler A, Hoffman P, et al. Comprehensive integration of single-cell data. *Cell* 2020;**177**(7): 1888–902.E21.
- Hafemeister C, Satija R. Normalization and variance stabilization of single-cell RNA-seq data using regularized negative binomial regression. *Genome Biol* 2019;**20**(1): 296.
- Schiebinger G, Shu J, Tabaka M, et al. Optimal-transport analysis of single-cell gene expression identifies developmental trajectories in reprogramming. *Cell* 2020;**176**(4): 928–43.
- Linderman GC, Rachh M, Hoskins JG, et al. Fast interpolation-based t-SNE for improved visualization of single-cell RNA-seq data. *Nat Methods* 2019;**16**(3):243–5.
- Seo E, Basu-Roy U, Gunaratne PH, et al. SOX2 regulates YAP1 to maintain stemness and determine cell fate in the osteo-adipo lineage. *Cell Rep* 2013;**3**(6):2075–87.

25. Hanieh H, Ahmed EA, Vishnubalaji R, et al. SOX4: epigenetic regulation and role in tumorigenesis. *Semin Cancer Biol* 2019, doi:10.1016/j.semcancer.2019.06.022.
26. Yu P, Nie Q, Tang C, et al. Nanog induced intermediate state in regulating stem cell differentiation and reprogramming. *BMC Syst Biol* 2018;12(1):22.
27. Cao J, Spielmann M, Qiu X, et al. The single-cell transcriptional landscape of mammalian organogenesis. *Nature* 2019;566(7745):496–502.
28. Cho H, Berger B, Peng J. Generalizable and scalable visualization of single-cell data using neural networks. *Cell Syst* 2020;7(2):185–91.
29. Hie B, Cho H, DeMeo B, et al. Geometric sketching compactly summarizes the single-cell transcriptomic landscape. *Cell Syst* 2020;8(6):483–93.
30. Setty M, Tadmor MD, Reich-Zeliger S, et al. Supporting data for “Wishbone identifies bifurcating developmental trajectories from single-cell data.” GitHub 2016. https://github.com/ManuSetty/wishbone/blob/master/data/sample_scseq_data.csv. Accessed 4 November 2020.
31. An S, Huang J, Wan L. Supporting data for “D-EE: a distributed software for visualizing intrinsic structure of large-scale single-cell data.” GigaScience Database 2020. <http://dx.doi.org/10.5524/100815>.