NETWORK EMBEDDINGS FOR DATA CLUSTERING, TRANSITION STATE IDENTIFICATION, AND ENERGY LANDSCAPE ANALYSIS

By

Paula Mercurio

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Mathematics – Doctor of Philosophy

2022

ABSTRACT

NETWORK EMBEDDINGS FOR DATA CLUSTERING, TRANSITION STATE IDENTIFICATION, AND ENERGY LANDSCAPE ANALYSIS

By

Paula Mercurio

Many chemical and biochemical systems can be intuitively modeled using networks. Due to the size and complexity of many biochemical networks, we require tools for efficient network analysis. Of particular interest are techniques that embed network vertices into vector spaces while preserving important properties of the original graph. In this article, we investigate several aspects of node embedding, propose a novel method of generating node embeddings, and explore applications to biochemical systems. We introduce a new method for generating low-dimensional node embeddings for directed graphs using random walk sampling and demonstrate the usefulness of this method for identifying transition states of stochastic chemical reaction systems, detecting relationships between nodes, and studying the structure of the original network.

In addition, we propose an efficient scheme for numerical implementation of network embedding based on deterministic computations of commute times rather than random walk trials. We develop a novel implementation of stochastic gradient descent (SGD) based on a low-dimensional sparse approximation of the original random walk on the graph, and show that this approach can improve the performance of node embeddings.

This method can be further extended for entropy-sensitive adaptive network embedding by incorporating principles from metadynamics and hierarchical network embedding, allowing for applications to the analysis of molecular structures. By adjusting the edge weights of the network by a Gaussian term, similarly to the metadynamics approach, we ensure that areas that have already been explored extensively by the random walk (i.e., the edges with the largest weights) will be de-emphasized over time, allowing additional iterations of the embedding process to reveal details about other areas of the graph. We show that this approach lends itself well to systems that are influenced by entropy or temperature effects and biochemical systems where the potential energy landscape depends on the system's configuration at a given time, either by itself or in conjunction with transition path theory. We demonstrate the effectiveness and performance of each of our methods on several datasets and biochemical examples.

For Mom and Dad.

ACKNOWLEDGEMENTS

First, I would like to thank my advisor, Di Liu, whose support and advice has made this entire work possible. His guidance has been indispensable throughout my time in graduate school, both with respect to our research and the development of my career. I am also very grateful to Guo-Wei Wei, Jiayu Zhou, Chichia Chiu, and Christina Chan, for the advice and help they have given me as my committee members over the years. I would also like to thank the entire math department at MSU, for giving me a wonderful graduate school experience.

I would like to thank my family, particularly my parents, Tillie and Dave Mercurio, whose unwavering encouragement has always given me support I can count on. Thank you to my sister Erin, who led the way and who has given me excellent advice throughout my time in graduate school, and to my brother Ryan, whose creativity and humor have helped make this journey much more fun.

Finally, I would like to thank my partner and best friend Whitney, whose companionship has always been vital. I'm looking forward to what comes next.

I would also like to acknowledge that parts of this work were supported by NSF-DMS 1720002, and that some of the databases of local minima and transition states were computed using the Pele, PATHSAMPLE, and disconnectionDPS softwares. The molecular graphics included in this research were produced with UCSF Chimera, developed by the Resource for Biocomputing, Visualization, and Informatics at the University of California, San Francisco, with support from NIH P41-GM103311.

The content from Chapter 1 is from [23]. The published version is available at AIMS Press via: https://doi.org/10.3934/mbe.2021046.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

CHAPTER 1

IDENTIFYING TRANSITION STATES OF CHEMICAL KINETIC SYSTEMS USING NETWORK EMBEDDING TECHNIQUES

1.1 Introduction

Networks provide a natural framework for many chemical and biochemical systems, including chemical kinetics and dynamical interactions between biomolecules. The possible stages of a reaction, for example, can be viewed as vertices in a directed graph. Frequently, real-world networks are high-dimensional and complex in structure, which leads to difficulties in interpretation and analysis. In order to take full advantage of the structural information contained in such networks, efficient and robust methods of effective network analysis are essential to produce low dimensional representations while retaining key information about the nodes of the graph.

Network analysis is especially useful for understanding behaviors of nonlinear chemical processes such as biochemical switches. Biochemical switches are systems of chemical reactions with multiple lower-energy steady states, or *metastable states*. Without stochastic perturbations, the system will spend all its time at the metastable states, where energy is minimized, while the presence of stochastic noise in the dynamics can lead to transitions between metastable states. Difficulties arise in the simulation of such systems as a result of the multiple time scales at work; transitions between steady states are typically rare, but occur rapidly.

The different possible states of the reaction, given by the varying concentrations of reactant and product species, can be viewed as the vertices of a weighted directed graph. For example, we may have an ‘initial’ state where only reactant species exist, a ‘final’ state where only product species remain, and any number of transitional states each representing a particular concentration of the different species. The edge weights of the graph can then be determined from the likelihoods of transitioning between any given pair of states. The progression of the reaction over time can then be viewed as a Markov jump process, where the jump probabilities are represented by the weights on the edges between each pair of nodes. Then, methods for network analysis can be used to identify interactions and pathways between the components of the biochemical switch. One objective of this paper is to suggest a method to analyze this Markov process using node embedding techniques. This paper will focus on the $A \to B$ problem: A and B are taken as the reactant and product states, respectively. The aim of this paper is to investigate the mechanism by which this transition occurs.

Recently, methods for analyzing networks through feature representation and graph embedding have received increasing attention. For an overview on this subject, a number of recent review papers are available [17, 5]. While much of this research has been focused on undirected graphs, the directed case has also been investigated, for example [9, 7, 51]. In particular, several methods have been proposed that utilize random walks for node embedding.

Node embedding (or *feature learning*) methods aim to represent each node of a given network as a vector in a low-dimensional continuous vector space, in such a way that information about the nodes and relationships between them are retained. Specifically, nodes that are similar to each other according to some measure in the original graph will also have similar representations in the embedding space. Usually, this means that the embedding will be chosen in a way that maximizes the inner products of embedding vectors for embedded nodes that are similar to each other. Compared with the original complex high-dimensional networks, these low dimensional continuous node representations have the benefit of being convenient to work with for downstream machine learning techniques.

Compared with alternative methods that embed edges or entire graphs instead of nodes, node embeddings are more adaptive for numerous tasks, including node classification, clustering, and link prediction [5]. Node classification is a process by which class labels are assigned to nodes based on a small sample of labeled nodes, where similar nodes have the same labels. Clustering algorithms group the representations of similar nodes together in the target vector space. Link prediction seeks to predict missing edges in an incomplete graph.

The recent interest in feature learning has led to a collection of different node embedding methods, which can be broadly classified [17] by the analytical and computational techniques employed such as matrix factorization, random walk sampling, and deep learning network. The matrix factorization based methods generate embeddings by first using a matrix to represent the relationships between nodes, which include the Laplacian matrix as in the Laplacian eigenmaps [3], the adjacency matrix as in locally linear embedding (LLE) [33] or graph factorization (GF) [1], etc. Though the above methods focus on the case of undirected methods, similar techniques for the directed case have also been suggested [9], where the development of the Laplacian matrix, a combinatorial Laplacian matrix, and Cheeger inequality allow the above approaches to be extended for directed graphs. Factorization based methods such as eigenvalue decomposition can apply for certain matrices (e.g., positive semidefinite), and additional difficulties may arise with scaling for large data sets.

Random walk methods are often used for large graphs, or graphs that cannot be observed in their entirety. The general idea involves first simulating random walks on a network, then using the output to infer relationships between nodes. Random walk methods can be used to study either undirected or directed graphs, although much of the previous work has focused on the undirected case. For example, DeepWalk [29] uses short unbiased random walks to find similarities between nodes, with node pairs that tend to co-occur in the same random walks having higher similarities. This method performs well for detecting *homophily* – similarity based on node adjacency and shared neighbors. Similar methods, such as node2vec [18], use biased random walks to capture similarity based on both homophily and *structural equivalence* – similarities in the nodes' structural roles in the graph. Both of these methods specifically address undirected graphs. A similar approach designed for the directed case is proposed in [7], which uses Markov random walks with transition probabilities given by ratios of edge weights to nodes' out degrees, together with the stationary distribution of the walks, to measure local relationships between nodes.

Theoretical frameworks for the study of transition events of biochemical processes include Transition State Theory (TST), Transition Path Sampling (TPS), Transition Path Theory (TPT) and Forward Flux Sampling (FFS). The main idea of TST is that in order for the system to move from the reactant state to the product state, the system must pass through a saddle point on the potential energy surface, known as a Transition State [48]. The TPS technique uses Monte Carlo sampling of transition paths to study the full collection of transition paths of a given Markov process [4]. The basic idea, similar to importance sampling, is to perform a random walk in trajectory space, biased so that important regions are visited more often. Transition Path Theory (TPT) studies reactive trajectories of a system by considering statistical properties such as rate functions and probability currents, which measure the average rates of transitions between states [47]. Forward flux sampling [2], designed specifically to capture rare switching events in biochemical networks, uses a series of interfaces between the initial and final states to calculate rate constants and generate transition paths. This method has the advantage that knowledge of the phase space density is not required. For the purpose of this paper, we will be using TPT, but other such techniques can fit into the method presented here as well.

The contributions of this paper are a method of using a random walk network embedding approach for node classification and clustering on directed graphs, as well as identification of transition states for the specific case of entropy effects. The method presented reduces the dimension of networks, thereby allowing analysis and interpretation of the system.

The remainder of this paper will be organized as follows. First, in Section 2 we will introduce some background regarding feature learning on networks, and provide a brief overview of some basic principles from TPT. Then Section 3 will introduce a method for node classification for directed graphs, based on random walk node embedding. Finally, in Section 4, we will study several examples, showing the effectiveness of the method on identifying transition states of entropic systems.

1.2 Background

In this paper, we focus on the mechanism by which a system such as a biochemical switch passes through its energy landscape following a *reactive trajectory*. A trajectory is said to be reactive if it leaves the reactant state A and later arrives at the product state B without first returning to state A. Such a trajectory can be viewed as a sequence of transitions between *metastable states* where the energy is locally minimized. In particular, we treat these sequences as Markov jump processes, and study the probability space of these reactive trajectories in order to identify and understand the transition events of the system. Though this paper will be utilizing the techniques of Transition Path Theory for this purpose, it is also possible to adopt other frameworks, such as Transition Path sampling, Forward Flux sampling, etc., to investigate the transition events of a system.

Consider a Markov jump process on a countable state space S. Let $\mathbb{L}$ represent the infinitesimal generator of the process, as defined in [24]. In other words, $\mathbb{L}$ is an $|S| \times |S|$ matrix with entries L_{uv} such that for $u, v \in S, u \neq v$, $L_{uv}\Delta t + o(\Delta t)$ represents the probability that a process that is in state u at time t will jump to state v during the time interval $[t, t + \Delta t]$. Then the entries of $\mathbb{L}$ are transition rates, satisfying

$$\begin{cases} L_{uv} \geq 0 & \forall\, u, v \in S,\ u \neq v, \\ \sum_{v \in S} L_{uv} = 0, & \forall\, u \in S. \end{cases} \tag{1.1}$$

Let $\{X(t)\}_{t\in\mathbb{R}}$ represent an equilibrium sample trajectory of the Markov process, such that $\{X(t)\}_{t\in\mathbb{R}}$ is right-continuous with left limits. At time t, let the probability distribution of this process be denoted by $\mu(t) := (\mathbb{P}(X(t) = u))^T_{u\in S}$. Then the time evolution of $\mu(t)$ follows the forward kolmogorov equation (or *master equation*)

$$\frac{d\mu}{dt} = \mu^T \mathbb{L}, \quad t \geq 0.$$

Denote the time-reversed process by $\{\tilde{X}(t)\}_{t\in\mathbb{R}}$, and the infinitesimal generator of this process by $\tilde{\mathbb{L}}$. The stationary probability distribution $\pi = (\pi_v)_{v\in S}$ of both $\{X(t)\}_{t\in\mathbb{R}}$ and $\{\tilde{X}(t)\}_{t\in\mathbb{R}}$ is given by the solution to

$$0 = \pi^T \mathbb{L}\ .$$

1.2.1 Network embedding for directed graphs

Networks can have large numbers of vertices and complex structures, which can lead to challenges in network analysis. To overcome these difficulties and improve understanding of these networks, the nodes of a given network can be embedded into a low-dimensional vector space, a process called *feature learning*. The techniques used in this paper to investigate transition states and pathways combine neural network-based and random walk-based methods for learning latent representations of a network's nodes, as discussed in [18, 29, 7]. The idea behind these methods is that the nodes in a given network $G = G(V, E)$ can be embedded into a low-dimensional vector space in such a way that nodes similar to each other according to some measure in the original graph will also have embeddings similar to each other in the embedding space.

Feature learning, or node embedding, requires three main components: an encoder function, a definition of node similarity and a loss function. The encoder function is the function that maps each node to a vector in the embedding space. To find the optimal embeddings, the encoder function can be chosen by minimizing the loss function so that the similarity between nodes in the original network corresponds as closely as possible to the similarity between their vector representations in the continuous space. Node similarity can have different interpretations, depending on the network and the task to be accomplished. For example, nodes that are connected, share neighbors, or share a common structural role in the original graph can be claimed to be similar, and to be embedded in the vector space by representations of close similarity. For the embedded vectors, a common approach is to use inner products as the similarity measure in the feature space–that is, if u and v are two nodes in a graph with representations $e(u)$ and $e(v)$ in a low-dimensional vector space, then $e(u)^T e(v)$ should approximate the similarity between representations $e(u)$ and $e(v)$.

1.2.1.1 Random walk approaches

One technique for finding node embeddings is through the use of random walks. This technique has been applied to undirected graphs in methods such as DeepWalk and node2vec [29, 18]. In these methods, the general approach is to use short random walks to determine similarities for each pair of nodes. The notion of node similarity is co-occurence within a random walk; two nodes are similar if there is a higher probability that a walk containing one node will also contain the other. The various methods using this approach differ in the implementation of this general idea: DeepWalk, for example, uses straightforward unbiased walks, while node2vec uses adjustable parameters to bias random walks toward breadth-first or depth-first sampling for a more flexible notion of similarity.

The first stage in the feature learning process is to simulate a certain number of random walks starting from each vertex. Using the results of these random walks, the neighbors of each node can be determined. The objective is to maximize the probability of observing the neighborhood of each node, conditioned on the vector representation of the node. In several of the random walk-based feature representation methods for undirected graphs, this is achieved through optimization of a Loss function of the form:

$$\max_{e} J(e) \;=\; \sum_{u \in V} \log Pr(N_S(u)|e(u)), \tag{1.2}$$

where V is the set of all vertices in the graph, $N_S(u)$ is the neighborhood of the node u, and $e(u)$ is the encoder function representing the embedded vector. If we assume that the probabilities of observing a particular neighbor are independent, we can have

$$Pr(N_S(u)|e(u)) = \prod_{n_i \in N_S(u)} Pr(n_i|e(u)). \tag{1.3}$$

Since these methods typically aim to maximize the inner products for neighboring nodes, they often utilize a softmax function to model each conditional probability:

$$Pr(n_i|e(u)) = \frac{\exp\Big(e(n_i) \cdot e(u)\Big)}{\sum_{v \in V} \exp\Big(e(v) \cdot e(u)\Big)} \,.$$

For networks where $|V|$ is large, techniques such as hierarchical softmax [29] or negative sampling [26] can be used to increase efficiency. To determine the optimal embedding function $e(u)$, the Loss function can be optimized using gradient descent with respect to the parameters of $e(u)$.

In [7], a random walk-based method specifically intended for directed graphs is proposed. The function to be optimized in this method, when embedding into $\mathbb{R}^1$, is

$$I = \sum_{u} \pi_u \sum_{v, u \to v} p(u,v)(y_u - y_v)^2,$$

where y_u is the one-dimensional embedding of u, π_u is the stationary probability for the node u and $p(u,v)$ represents the random walk's transition probability from u to v. The transition probabilities are calculated according to

$$p(u,v) = w(u,v) / \sum_{k, u \to k} w(u,k),$$

where $w(u, v)$ is the weight on the edge (u, v). Note that this method, though it uses transition probabilities of a random walk, does not actually require simulation of random walks, since all the summations can be done deterministically.

According to the spectral graph theory [9], a circulation on a directed graph is a function F such that, for each $u \in V$, $\sum_{u,u \to v} F(u, v) = \sum_{w,v \to w} F(v, w)$. The Laplacian of a directed graph is therefore defined as

$$\mathcal{L} = I - \frac{\mathbf{\Phi}^{1/2}\mathbf{P}\mathbf{\Phi}^{-1/2} + \mathbf{\Phi}^{-1/2}\mathbf{P}\mathbf{\Phi}^{1/2}}{2},$$

where $\mathbf{\Phi}$ is a diagonal matrix with diagonal entries given by $\mathbf{\Phi}_{v,v} = \phi(v) = \sum_{u,u \to v} F(u, v)$ for some circulation function F. The following definition of the *combinatorial Laplacian* is also due to Chung [9]:

$$\mathbb{L} = \mathbf{\Phi} - \frac{\mathbf{\Phi}\mathbf{P} + \mathbf{P}^T\mathbf{\Phi}}{2},$$

where $\mathbf{P}$ is the transition matrix, i.e., its elements are given by $P_{uv} = p(u, v)$, and $\mathbf{\Phi}$ is the diagonal matrix of the stationary distribution, i.e., $\mathbf{\Phi} = diag(\pi_1, \ldots, \pi_n)$. Note that the combinatorial Laplacian is symmetric and semi-positive definite. It is shown in [7] that

$$\sum_u \pi_u \sum_{v,u \to v} p(u, v)(y_u - y_v)^2 = 2\mathbf{y}^T\mathbb{L}\mathbf{y}$$

where $\mathbf{y} = (y_1, y_2, \ldots, y_n)^T$. This result demonstrates that this method is analogous to the Laplacian-based methods for undirected graphs, e.g., Laplacian eigenmaps [17].

Random walk methods have been shown to perform well compared to other methods, and can be useful for a variety of different notions of node similarity. Previous work has shown them to be robust, efficient, and capable of completing a diverse range of tasks, including node classification, link prediction, clustering, and more [17, 5, 29, 18]. Because they do not examine the entire graph at once, these methods are also useful for very large networks and networks that cannot be observed in their entirety.

1.2.1.2 Neural network approaches

A third class of graph embedding methods involves the use of deep learning techniques, including neural networks [5, 17], e.g., Graph Neural Network (GNN) [5, 35]. Such methods have been shown to perform well, particularly for dimension reduction and identifying nonlinear relationships among data. Random walk based methods, such as DeepWalk, can incorporate deep learning algorithms like SkipGram for graph embedding, where random walks serve to provide neighborhood information as input.

In general, neural network methods typically work by assigning a weight to each node representation, and then combining those terms via a transfer function. This result is then used as the input for an activation function, such as a sigmoid function, i.e., $\sigma(x) = 1/(1 + \exp(-x))$. This process may then repeat for multiple layers. In a neural network method, the parameters to be optimized are the weights, which are updated after each layer, typically by gradient descent. In the GNN [35], the objective function to be minimized is of the form

$$W = \sum_i \sum_j (t_{i,j} - \varphi_w(G_i, n_{i,j}))^2,$$

where G_i is the learning set, the $n_{i,j}$ and $t_{i,j}$ are the nodes and target outputs, and the goal is to choose the parameter w in such a way that φ closely approximates the target outputs. This optimization occurs through gradient descent, where the gradient is computed with respect to the weights w.

1.2.2 Transition path theory

In order to identify the transition paths of a given Markov process, we will adopt the framework of Transition Path Theory (TPT). The following notations are mostly from [24]. Consider a state space S, for an initial state $A \in S$ and final state $B \in S$, a trajectory $X(t)$ is said to be reactive if it begins from state A and reaches state B before returning to state A.

To determine whether a given reaction path is reactive, we will need the following forward and backward committor functions. For each $i \in S$, the forward committor q_i^+ is the probability that a process initially at state i will reach state B before it reaches state A. Similarly, the discrete backward committor q_i^- denotes the probability that a process arriving at state i was more recently in state A rather than in state B. The forward committors satisfy the following Dirichlet equation

$$\begin{aligned} \sum_{j \in S} L_{ij} q_j^+ &= 0 \qquad && \forall i \in (A \cup B)^C \\ q_i^+ &= 0 && \forall i \in A \\ q_i^+ &= 1 && \forall i \in B, \end{aligned}$$

and similar equations are satisfied by backward committors.

The probability current, or flux, of reactive trajectories gives the average rate at which reactive paths transition from state to state. For trajectories from A to B, the probability current is defined for all $i \neq j$ such that

$$\begin{aligned} f_{ij}^{AB} &= \lim_{s \to 0+} \frac{1}{s} \lim_{T \to \infty} \frac{1}{2T} \int_{-T}^{T} \mathbf{1}_{\{i\}}(X(t)) \mathbf{1}_{\{j\}}(X(t+s)) \times \sum_{n \in \mathbb{Z}} \mathbf{1}_{(-\infty, t_n^B]}(t) \mathbf{1}_{[t_n^A, \infty)}(t+s) dt \\ &= f_{ij}^{AB}, \end{aligned}$$

where $\mathbf{1}_{(a,b)}$ denotes the indicator function on the interval (a, b). Also, for all $i \in S$, $f_{ii}^{AB} = 0$. It can be shown that

$$f_{ij}^{AB} = \begin{cases} \pi_i q_i^- L_{ij} q_j^+, & \text{if } i \neq j \\ 0 & \text{if } i = j. \end{cases}$$

Since we are primarily interested in the flow from A to B, and the process can move in either direction between two adjacent nodes on the path, the following effective current accounts for the net average number of reactive paths that jump from i to j per time unit:

$$f_{ij}^+ = \max(f_{ij}^{AB} - f_{ji}^{AB}, 0).$$

We can also define the total effective current for a node i:

$$C_i^+ = \sum_{j:(i,j) \in E} f_{ij}^+.$$

In [14], the effective current is generalized to a connected subnetwork of nodes Ω such that

$$C^+(\Omega) = \sum_{i \in \partial\Omega} C_i^+,$$

which leads to the following definition of transition states of a time reversible process as subsets of the state space:

$$TS = \lim_{\sigma \to 0} \operatorname*{argmax}_{\Omega} \{C^+(\Omega) \exp\left(-(q^+ - 0.5)^2/\sigma^2\right)\}$$

For the non-time reversible case, transition states can be identified similarly, replacing the exponential argument $(-(q^+ - 0.5)^2/\sigma^2)$ with $(-(q^+ - 0.5)^2 + (q^- - 0.5)^2)/\sigma^2$. The above definition gives the flexibility of identifying transition states as subsets of the state space, which will be very useful when dealing with complex dynamics such as entropy effects.

1.3 Identifying transition states using network embedding

We now introduce a method to identify transition states and paths of Markov processes, by random walk based network embedding techniques for directed networks. Consider a Markov jump process in a countable state space S. Using the infinitesimal generator $\mathbf{L}$ for the process, we can calculate the stationary probability distribution π and the forward and backward committors, based on which probability current and effective currents f_{ij}^{AB} and f_{ij}^+ can then be computed. Once these quantities have been obtained, the discretized state space can be viewed as a weighted, directed graph $G(V, E)$. The nodes $v \in V$ are the grid points representing states of the system, and each pair of adjacent nodes are connected by an edge $(u, v) \in E$ with weights given by the effective probability current, such that the direction of the edge will be determined by the sign of the effective current.

Once G is constructed, we can apply feature learning techniques for node classification to identify transition states. Random walk trials for similarities between nodes will start from each node in the space, using the edge weights as transition probabilities. The outputs of the random walks can then be used to compute experimental “neighbor probabilities", i.e., the probability that a random walk of length d starting at node u will contain node v. If the neighbor probability for a node pair (u, v) is sufficiently high, then the node v is claimed as a neighbor of u. Note that this is a directed process; v can be a neighbor of u even if u is not a neighbor of v.

After finding these conditional probabilities and identifying the neighborhoods of each node, the next step is to maximize an objective function in order to solve for the node embeddings. As in the node2vec and DeepWalk methods for the undirected case, the conditional probabilities are modeled using softmax functions. In order to adapt this technique for the case of directed graphs, we will use a modified version of the objective function:

$$V[e] = \sum_{u} \pi_u \sum_{v \in N(u)} Pr(v|e(u)). \tag{1.4}$$

Optimizing this function will give the encoder function e^{opt} that maximizes the probability of the neighborhood of each node, with the most relevant nodes having the most influence on the feature representations.

In the original form of the objective function (1.4) for the undirected graph methods described above, the softmax functions depend only on whether nodes are neighbors, and is independent of the probability (frequency) that a particular neighbor will show up in a random walk trial. In other words, all neighbors are treated equally, while in reality some node pairs are more likely to co-occur than others. To address this, the objective function used here incorporates the neighboring probabilities, denoted $NP(\cdot,\cdot)$ such that

$$Pr(n|e(u)) = \frac{\exp(e(n) \cdot e(u))NP(n,u)}{\sum_{v \in V} \exp(e(v) \cdot e(u))NP(v,u)}.$$

Assuming $e(u) = \mathbf{E_2}u$, for $\mathbf{E_2}$ being a $2 \times d$ matrix, the objective defined above can then be maximized using gradient ascent to obtain the matrix ${\mathbf{E_2}}^{opt}$ and therefore the feature embeddings. In practice, the linear assumption on the map from nodes to embeddings appears to be insufficiently flexible, particularly for higher-dimensional problems. To improve upon this, a neural network can be incorporated into the feature learning process, with the sigmoid function $\sigma(u) = 1/(1+\exp(-e(u)))$ where $e(u)$ is the representation in the embedding space of the node u. The current algorithm uses a relatively small number of layers; raising this number results in node representations that are clustered more closely to their neighbors.

Algorithm 1.1 A random walk simulation method for directed graphs, given the transition matrix T.

```
for $u \in V$ do
  $\text{walk}(1) = u$
  for $\text{step} = 1 : \text{walklength}$ do
    for $k = 2 : d$ do
      $\text{total} = \sum_{v,\text{walk}(k-1)\to v} \mathbf{t}(\text{walk}(k-1), v)$
      for $v : \text{walk}(k-1) \to v$ do
        $P(\text{walk}(k-1), v) = \mathbf{t}(\text{walk}(k-1), v)/\text{total}$
      end for
      Use probability $P(\text{walk}(k-1), \cdot)$ to choose $\text{walk}(k) = v$ for some neighbor of $\text{walk}(k-1)$.
      Set $\text{counter}(\text{walk}(k), u) = \text{counter}(\text{walk}(k), u) + 1$.
    end for
  end for
end for
for $u, v \in V$ do
  $\text{neighborprobability}(u, v) = \text{counter}(u, v)/\sum_k \text{counter}(u, k)$
end for
```

To obtain transition states for a particular process, we will examine the similarities between the node representations of each state and the representation of metastable state A. Here, by “similarities" we mean the conditional probabilities given by the softmax units discussed previously. By construction, these values will naturally be small for nodes that are further from node A, due to hopping distances that are longer than the length of the random walks. So, to find the transition states, we combine similarities of node pairs with shorter hopping distances via matrix propagation, e.g.,

$$\text{sim}(A, u) = \sum_{A\to v, v\to u} \text{sim}(A, v)\text{sim}(v, u). \tag{1.5}$$

Note that a node u for which this value is high is a higher-order neighbor of the reactant node A; therefore, such a node will have a higher probability of lying along a transition path. We then define transition states as nodes with high probability of lying on a transition path, which are not direct neighbors of the reactant or product states.

1.4 Examples and results

Next, we will illustrate the suggested method on several simple examples. In the examples presented here, we have used 100 random walk trials, each of length 9 steps. Smaller numbers of trials increase

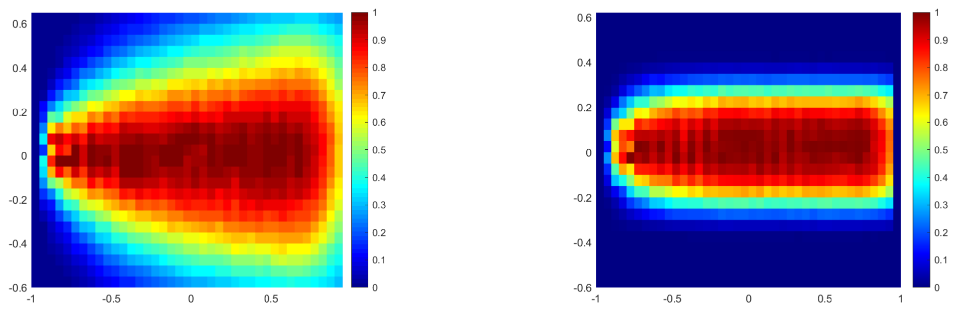

Figure 1.1 Diffusion process with energy barrier: plot assigning colors to each node in the discretized domain Ω, according to each node's similarity to the starting node A. Higher similarity values correspond to higher dot products between the two-dimensional embeddings of the nodes. The left plot corresponds to a parameter value of $\varepsilon = 0.01$, and the right plot corresponds to $\varepsilon = 1$.

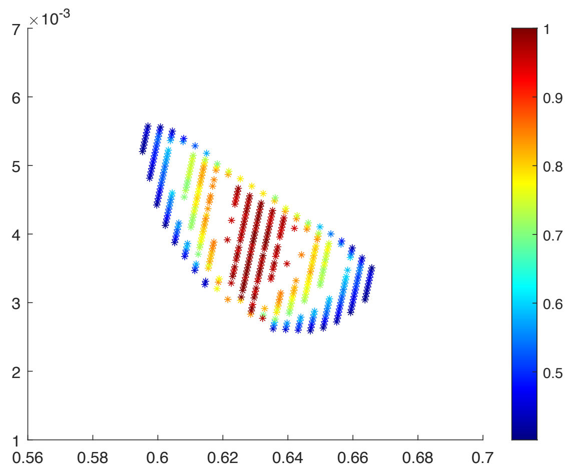

Figure 1.2 Feature representations of nodes with high similarity to the reactant state A for the 2D diffusion process with energy barrier, with $\varepsilon = 0.01$.

the impact of noise in the results. Due to the matrix propagation technique described above, changes to the walk length parameter do not have a noticeable affect.

1.4.1 Diffusion process with energy barrier

First we will consider a two-dimensional diffusion process $(x(t), y(t))$ representing the position of a particle traversing a potential field with potential V. This process follows the SDE

$$\frac{dx}{dt} = -\frac{dV}{dx} + \frac{dW_1(t)}{dt},$$
$$\frac{dy}{dt} = -\frac{dV}{dy} + \frac{dW_2(t)}{dt},$$

where V is the potential

$$V = (x^2 - 1)^2 + \varepsilon y^4, \tag{1.6}$$

and $W_1(t)$ and $W_2(t)$ are independent Brownian motion processes.

This potential V has two local minima at (-1,0) and (1,0), as well as a saddle point at the origin. As a result, the diffusion process $(x(t), y(t))$ will have metastable states at these energy minima. Here I will take (-1,0) to be the reactant state A, and (1,0) to be the product state B. The diffusion process can be approximated using a Markov (birth-death) jump process on a discrete state space. For this example, we will study the diffusion process on the domain $\Omega = [-1, 1] \times [-0.75, 0.75]$ by examining the Markov jump process on a grid $D = ((-1 + h\mathbb{Z}) \times (-0.75 + h\mathbb{Z})) \cap \Omega$, with $h = 0.05$.

In order to examine the behavior of this Markov process, we should first compute the probability currents for the process. The infinitesimal transition matrix L can be found by using the jump rates for the birth-death process L_{uv} for each pair of adjacent nodes u, v [15]. We define the following constants for a state (x, y) with $x \in (-1, 1)$:

$$k_x^+(x, y) = \frac{1}{2h^2} - \frac{1}{2h}\frac{dV}{dx}(x - h),$$
$$k_x^-(x, y) = \frac{1}{2h^2} + \frac{1}{2h}\frac{dV}{dx}(x + h),$$
$$k_y^+(x, y) = \frac{1}{2h^2} - \frac{1}{2h}\frac{dV}{dy}(y - h),$$
$$k_y^-(x, y) = \frac{1}{2h^2} + \frac{1}{2h}\frac{dV}{dy}(y + h).$$

Also let $k_x^+(x+h,y) = 1/h$ if $x = -1$, $k_x^+(x+h,y) = 0$ if $x = 1$, and $k_x^-(x-h,y) = 0$, $k_x^-(x-h,y) = 1/h$ if $x = -1$ or $x = 1$ respectively, and similarly for k_y^+ and k_y^-. Then the infinitesimal generator can be expressed in terms of its action on a test function f such that

$$\begin{aligned}(\mathbf{L}f)(x,y) =& k_x^+(x+h,y)(f(x+h,y)-f(x,y)) + k_x^-(x-h,y)(f(x-h,y)-f(x,y)) \\ &+ k_y^+(x,y+h)(f(x,y+h)-f(x,y)) + k_y^-(x,y-h)(f(x,y-h)-f(x,y)).\end{aligned}$$

Figure 1.1 assigns each point in the grid a color based on its similarity to the node of $A = (-1,0)$, for epsilon values $\varepsilon = .01$ and $\varepsilon = 1$ in Eq (1.6), respectively. To account for the fact that nodes with greater hopping distances from A will have lower similarities to this node, we propagate the similarity matrix according to (1.5) in order to assign similarities to more distant nodes. The red area passing from A to B represents the region that reactive trajectories will pass through with the highest probability for each value of epsilon.

In Figure 1.2, the node embeddings $(u,v) = e_{opt}(x,y)$ corresponding to a subset of nodes with higher similarity values to A are shown, for the case where $\varepsilon = 0.01$, where e_{opt} is the optimal linear encoding. Notice that in this figure, the embeddings for nodes with the highest probability of occurring in a reactive trajectory are clustered together, while nodes with lower probabilities tend to be grouped with other nodes with similar probabilities; more specifically, nodes with similarity values in the range $0.5 - 0.6$ have embeddings clustered near the upper left or lower right of the figure, while nodes with values from 0.7–0.8 will have embeddings which appear in one of the two orange clusters.

In the context of the diffusion process with the potential V, the *entropy effect* refers to changes in the system's observed dynamics in response to the change of the parameter ε. That is, as ε is decreased, dV/dy shrinks and therefore the negative term of dy/dt becomes small. As a result, the shape and size of the saddle point located at the origin is altered, and we can expect the y-coordinate to take on a greater variety of values with higher probabilities in this case. In particular, around the origin, smaller values of ε should result in higher similarities between the nodes above and below the origin, since in this case the process is more likely to move up and down. For larger values of ε, the process is expected to move straight forward from A to B, with a smaller probability of moving up or down.

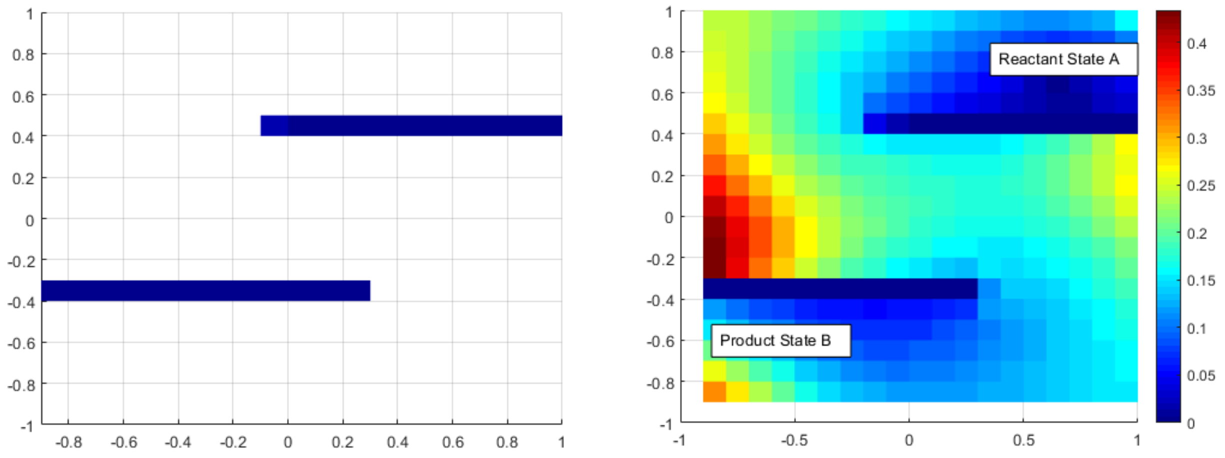

Figure 1.3 Diffusion process with entropic bottleneck: surface plot of the similarity function between the reactant state A and nodes in Ω. Here, the discretization with $h = 0.1$ is used.

1.4.2 Diffusion process with barriers

Next we consider a pure diffusion process ($V(x, y) = 0$) on the domain $\Omega = [-1, 1] \times [-1, 1]$ with two barriers as depicted in Figure 3. This example was also discussed in [25]. Here we take the node at $(0.6, 0.6)$ to be the initial state A, and $(-0.6, -0.6)$ as the final state B. The particle spends most of its time between the reactant and product states, in the region between the barriers ($[-1, 1] \times [-0.4, 0.4]$). The only factor affecting the probability that the particle will reach B before returning to A is its current distance to B, since there are no energetic obstacles to overcome. In order to travel from A to B, the process must find its way past the obstacles by chance, thereby overcoming an *entropic barrier*.

Figure 4 shows the node embeddings for a subset of nodes with high similarity values to reactant A. Note that the color scheme used in this figure corresponds to the color scheme in Figure 3; node embeddings that are colored red in the scatter plot correspond to red areas in the surface plot. In Figure 4, node embeddings again tend to be grouped by the corresponding values of the similarity function: nodes that are very similar to the initial state A (similarity function value > 0.9) appear in the red cluster toward the lower right. Other clusters of different colors correspond to groups of

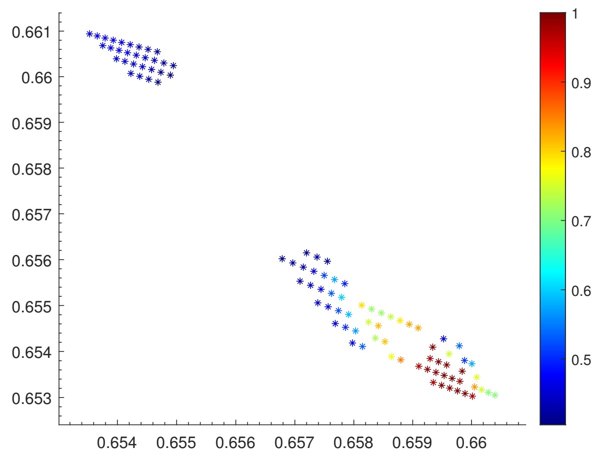

Figure 1.4 Feature representations of nodes with high similarity to the reactant state A for the entropic diffusion process.

nodes with lower probabilities of appearing in a reactive trajectory. The yellow cluster contains embeddings of nodes with similarity function values approximately 0.7–0.8, while the blue cluster corresponds to nodes with similarity values near 0.5–0.6.

1.4.3 Three-dimensional toggle switch

Now we will consider a higher-dimension example, a stochastic model of a 3D genetic toggle switch consisting of three genes that each inhibit the others' expression [14]. We consider the production and degradation of the three gene products, S_1, S_2, and S_3:

$$* \underset{\alpha_4}{\overset{\alpha_1}{\rightleftharpoons}} S_1, * \underset{\alpha_5}{\overset{\alpha_2}{\rightleftharpoons}} S_2, * \underset{\alpha_6}{\overset{\alpha_3}{\rightleftharpoons}} S_3,$$

where the parameters α_i are defined:

$$\alpha_1 = \frac{c_{11}}{(65 + x_2^2)(65 + x_3^2)}, \qquad \alpha_2 = \frac{c_{12}}{(65 + x_1^2)(65 + x_3^2)},$$

$$\alpha_3 = \frac{c_{13}}{(65 + x_1^2)(65 + x_2^2)}, \qquad \alpha_4 = c_4 x_1,$$

$$\alpha_5 = c_5 x_2, \qquad \alpha_6 = c_6 x_3,$$

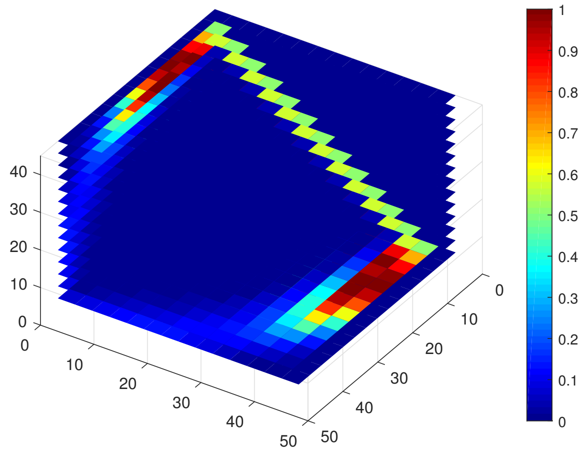

Figure 1.5 Surface plot of the similarities to A of grid points in the domain Ω for the 3-D Toggle switch model. Results are for a $15 \times 15 \times 15$ discretization ($h = 3$).

with $c_{11} = 2112.5$, $c_{12} = 845$, $c_{13} = 4225$, $c_4 = 0.0125$, $c_5 = 0.005$, and $c_6 = 0.025$. From the stationary probability distribution, it can be seen that this model has three metastable states:

$$A = \{x \in S | 35 \leq x_1 \leq 45, 0 \leq x_2 \leq 4, 0 \leq x_3 \leq 4\},$$

$$B = \{x \in S | 35 \leq x_2 \leq 45, 0 \leq x_1 \leq 4, 0 \leq x_3 \leq 4\},$$

$$C = \{x \in S | 35 \leq x_3 \leq 45, 0 \leq x_2 \leq 4, 0 \leq x_1 \leq 4\}.$$

Here we will choose A to be the reactant state and B to be the product state.

Figure 5 assigns colors to each node in $\Omega = \{ih, jh, kh | i, j, k \in \mathbb{Z}\} \cap [0, 45] \times [0, 45] \times [0, 45]$ based on their similarity to nodes in A, propagating the similarity matrix for nodes with larger hopping distance from A as described previously. This figure shows two potential transition states for this system. A reactive trajectory from A to B will be most likely to pass directly from A to B along the higher-probability trajectory, represented in red. However, such a trajectory may instead reach B after travelling through state C, as indicated by the fainter region connecting A and B via C.

Below, Figure 6 shows the two-dimensional node embeddings in $\mathbb{R}^2$, with the embeddings of nodes with zero similarity to A omitted for clarity. There are several distinct clusters present in Figure 6, each representing a cluster of similar nodes. The nodes with the highest similarities to A

after propagation of the similarity matrix (represented in red) are shown in the cluster to the lower left of the figure. Hence, the nodes in this cluster represent the nodes expected to appear in the transition path.

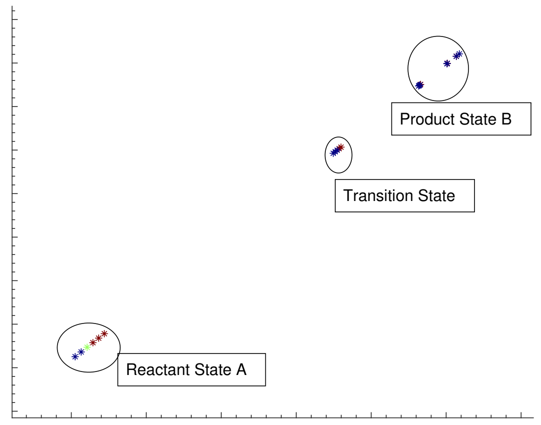

Figure 1.6 Feature representations of nodes with high similarity to the reactant state A for the 3D toggle switch.

1.4.4 Stochastic virus model

Now we will consider a higher-dimension example, a 3D stochastic model for virus propagation consisting of three species which are necessary for virus production [38]. The three species involved in this system are the template (*tem*), viral genome (*gen*), and structural (*struct*) proteins of the virus.

The cellular concentrations of these proteins are controlled by six reactions: producing *tem* from *gen*, using *tem* as a catalyst to produce *struct* and *gen*, degradation of *tem* and *struct*, and propagation of the virus using *struct* and *gen* such that

$$\begin{aligned} &[gen] \xrightarrow{k_1} [tem]\,, && [tem] \xrightarrow{k_2} \emptyset, \\ &[tem] \xrightarrow{k_3} [gen]\,, && [gen] + [struct] \xrightarrow{k_4} \emptyset, \\ &[tem] \xrightarrow{k_5} [struct]\,, && [struct] \xrightarrow{k_6} \emptyset, \end{aligned}$$

where we adopt $k_1 = 0.25, k_2 = 0.25, k_3 = 1.0, k_4 = 7.5 \times 10^{-6}, k_5 = 1000,$ and $k_6 = 1.99$. From [38] we know there are two steady states for this system: an unstable trivial solution at

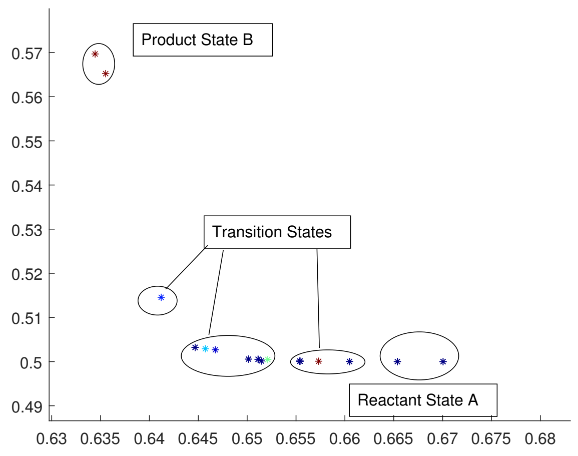

Figure 1.7 Feature representations of nodes with high similarity to the initial state $A = (0, 0, 0)$ for the virus propagation example.

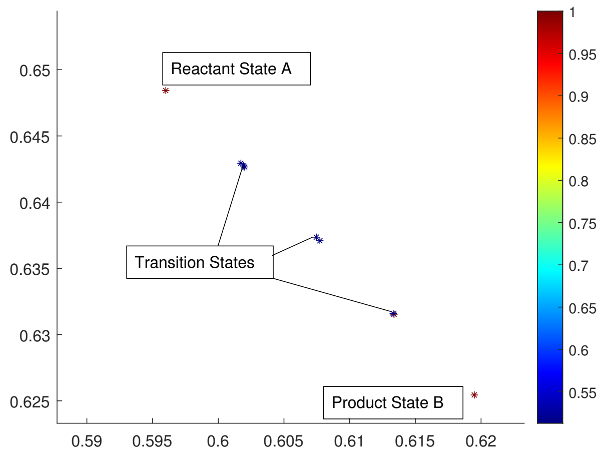

Figure 1.8 Feature representations for the virus propagation example, with a logarithmic scale on the x-axis.

$tem = gen = struct = 0$, and a stable steady state, which with the given parameter values is $tem = 30, gen = 100, struct = 12000$. Ignoring the stochastic effects, as a macro scale limit, this system can be modeled by the ODEs:

$$\begin{aligned}
\frac{d[tem]}{dt} &= k_1[gen] - k_2[tem], \\
\frac{d[gen]}{dt} &= k_3[tem] - k_4[gen][struct] - k_1[gen], \\
\frac{d[struct]}{dt} &= k_5[tem] - k_6[struct] - k_4[gen][struct].
\end{aligned}$$

For a stochastic perspective, we model the system by a Markov process, where we follow the First Reactions method of the Gillespie algorithm[16], which at each time step assumes that the reaction that could take place in the shortest amount of time will occur next.

From the upper left to the lower right, these clusters contain the representations for nodes near approximately $(15, 0, 1000)$, $(30, 66.7, 4000)$ and $(30, 100, 8000)$. Direct simulations confirm that these are transition states: reactive trajectories will tend to pass through nodes in each of these clusters. Applying the feature learning method to this system produces a two-dimensional feature representation for each three-dimensional node $(tem, gen, struct)$. Here, we take state A to be the origin and $B = (30, 100, 12000)$. Figure 7 plots the two-dimensional feature representations for this system. In this figure, the nodes with higher similarities to the initial state (represented in red) and those with lower such similarities (represented in blue) appear in separate clusters. Figure 8 plots these feature representations with a logarithmic scale on the x-axis to better display the clusters.

Five distinct clusters are discernible in Figure 1.8: the red node in the top left is the feature representation for the reactant state, the red node at the bottom right is the representation of the product state, and the three clusters between each represent a transition state that reactive trajectories pass through while traveling from A to B. From the upper left to the lower right, these three clusters contain the representations for nodes near approximately $(15, 0, 1000)$, $(30, 66.7, 4000)$ and $(30, 100, 8000)$. Direct simulations confirm that these are transition states: reactive trajectories will tend to pass through nodes in each of these clusters.

1.4.5 E. Coli σ-32 heat response circuit

Finally we will address a higher-dimension example, to illustrate the effectiveness of the method on reducing the dimension of a graph. Consider the E. coli σ-32 stress circuit, a network of regulatory pathways controlling the σ-32 protein, which is essential in the E. coli response to heat shock [37]. The systems consists of 10 processes:

$$
\begin{aligned}
&FtsH \xrightarrow{k_1} \emptyset, && E\sigma^{32} \xrightarrow{k_2} FtsH,\\
&GroEL \xrightarrow{k_3} \emptyset, && E\sigma^{32} \xrightarrow{k_4} GroEL,\\
&\sigma^{32} + J_{comp} \xrightarrow{k_5} \sigma^{32}\text{-}J_{comp}, && \sigma^{32}\text{-}J_{comp} \xrightarrow{k_6} J_{comp} + \sigma^{32},\\
&\sigma^{32}\text{-}J_{comp} \xrightarrow{k_7} FtsH, && \emptyset \xrightarrow{k_8} \sigma^{32},\\
&E + \sigma^{32} \xrightarrow{k_9} E\text{-}\sigma^{32}, && E\text{-}\sigma^{32} \xrightarrow{k_10} E + \sigma^{32}.
\end{aligned}
$$

In this context, FtsH and GroEL are stress response proteins, E is a holoenzyme, and J_{comp} (or J-complex) represents several chaperone proteins that are lumped as a simplification. Then E-σ^{32} denotes the protein complex formed when E binds to σ^{32}, which catalyzes downstream synthesis reactions. σ^{32} is a product of translation, which we assume to occur at a rate corresponding to $k_8 = 0.007$. It can also associate and dissociate of the J-complex. In accordance with [37] the other rate constants are taken to be: $k_1 = 7.4 \times 10^{-11}, k_2 = 4.41 \times 10^{6}, k_3 = 1.80 \times 10^{-8}, k_4 = 5.69 \times 10^{6}, k_5 = 3.27 \times 10^{5}, k_6 = 4.4 \times 10^{-4}, k_7 = 1.28 \times 10^{3}, k_9 = 0.7$, and $k_{10} = 0.13$.

For the above reaction rates, this system has a metastable state where the concentrations of FtsH, GroEL, σ^{32}, and J_{comp} are approximately 600, and the concentrations of the other three species are approximately 1500. Another metastable state occurs when the concentrations of FtsH, GroEL,σ^{32}, and J_{comp} are approximately 800. Here we take the former to be initial state A and the latter to be the product state B. Following a procedure similar to that in the above virus propagation example, three-dimensional representations are generated for each node. The plot of these node representations is given in Figure 9. The color scale in this figure is determined by the value of the similarity function between each node and the node A, with the most similar nodes shown in red.

Nodes with lower similarities have been omitted for clarity. Four distinct clusters can be seen in Figure 9.

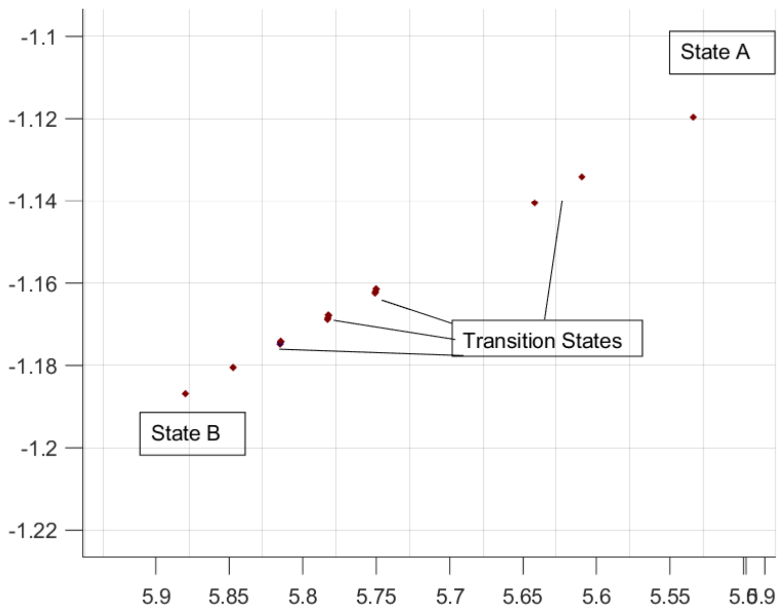

Figure 1.9 Feature representations of nodes with high similarity to the initial state, when initial state is taken to be $A = (600, 600, 600, 600, 1500, 1500, 1500)$, for the E. Coli heat response circuit model.

1.5 Conclusions

In this paper, we presented a method for analyzing metastable chemical reaction systems via feature learning on directed networks using random walk sampling. We have shown how this method may be used to identify transition states of Markov jump processes by interpreting such processes in terms of directed networks and taking advantage of Transition Path Theory. We have illustrated the efficacy of this method through several low-dimensional examples involving various energetic and entropic barriers. As noted above, for more complex, realistic examples, the method can be used for dimensional reduction, enabling the extraction and analysis of high-dimensional graph information. Further work will be necessary to fully understand this approach from a probabilistic standpoint. Another direction for future work is improving efficiency and development of faster algorithms. However, this method still forms the groundwork for a potential new way of analyzing directed networks and jump processes, particularly in the context of chemical kinetics.

CHAPTER 2

EFFICIENT NETWORK EMBEDDING BASED ON SPARSE APPROXIMATION OF A RANDOM WALK

2.1 Introduction

Graphs provide a versatile and intuitive representation for many real world situations. A wide variety of relational data, e.g., social networks, protein-protein interaction networks, biochemical systems, citation networks, and many others, can be interpreted as graphs. As a result, methods for analyzing these networks have consistently garnered much interest.

Network embedding, or *feature learning*, which provides a major framework for network analysis, seeks to map each node in a given graph to a point in a low-dimensional vector space such that relational information from the original graph is preserved. In particular, the embeddings of nodes that are closely related in the original graph (e.g., those connected by an edge, or with mutual neighbors) will be close together in the embedding space. In many machine learning applications, the lower-dimensional vectors are much easier and more convenient than the original network representations.

Although there also exist methods for embedding edges or whole networks, here we focus on node embedding methods. These frequently have the benefit of being more adaptive and useful for a variety of applications, such as node classification, clustering, and link prediction [5]. Node classification assigns class labels to unlabeled nodes based on a small sample of labeled nodes, while clustering algorithms group the representations of similar nodes together in the target vector space, and link prediction is used to predict edges based on a sample of data containing edge information.

Most node embedding methods can be classified based on the general techniques used to compute representations. As seen in [17], the three major categories are methods based on matrix factorization, random walk sampling, and deep learning techniques. Matrix factorization methods involve representing network information in a matrix (e.g. an adjacency matrix or Laplacian), then factoring or decomposing that matrix to determine embeddings. Examples include Laplacian eigenmaps [3], locally linear embedding (LLE) [33], and graph factorization [1]. Factorization methods are deterministic and often perform well on small graphs, but may be impractical on very large graphs. Some factorization methods can only be applied to matrices with specific properties; methods involving eigenvalue decompositions, for instance, require the matrix to be positive semidefinite.

A benefit of random walk methods is that these methods are not typically subject to these restrictions, and can easily be applied to large networks. In general, these methods involve collecting information about node-node relationships by running short random walk trials on the network, then encoding this information into embeddings via gradient descent. DeepWalk [29] and node2vec [18] are two well-known methods of this type for undirected graphs, and a random walk approach for the directed case is proposed in [7]. In DeepWalk [29], closely-related nodes are embedded near each other based on the frequency of co-occurrence within unbiased, fixed length random walks on the graph. One method that builds on this idea is node2vec [18], which replaces the simple unbiased random walks with walks influenced by parameters that control the walks' tendency to either backtrack and stay near the starting node, or explore higher-degree neighbors of the starting node.

Here, we propose another improved implementation of SGD-based random walk methods, based instead upon a sparse approximation of the random walk on the graph. We use this sparse approximation to deterministically find embeddings optimizing the *commute time* [19], then finally apply SGD to minimize cross-entropy loss. This approach achieves efficiency by avoiding Monte Carlo simulations of random walks, instead leveraging commute time, which takes into account all paths of arbitrary length between nodes.

The *commute time* is defined as the expected time for a random walk to travel from one node u to another v, and then back to the starting node u. It can also be viewed as an integration over time t of the *diffusion distance* introduced in [10], which is defined through the probability of the random walk traveling from u to v in fixed time t. Hence, the diffusion distance sums over all paths of finite time t between the pair of nodes (u, v). One advantage of utilizing commute time rather than diffusion distance is that it removes the need to choose an appropriate value for the parameter of time t.

Numerically, the commute time between two nodes can be computed from the Green function. Through the Schultz method, one can compute an efficient approximation of the Green function via a short product involving dyadic powers of the random walk's transition matrix. The *diffusion wavelets* algorithm, introduced in [11], offers one way to compute increasingly low-dimensional, compressed representations for these dyadic powers. Diffusion wavelets can also be used for network embedding, as discussed in [44]. By construction, at each scale represented by a dyadic power of the diffusion operator, the diffusion wavelets algorithm produces a basis of scaling functions that span the same space as the eigenvectors corresponding to the largest eigenvalues. As a result, these scaling functions can be used to produce embeddings that are equivalent up to a rotation to those produced by kernel methods. Alternatively, the low-dimensional approximations of the dyadic powers (and subsequently the Green function) can be computed from the Markov matrix's largest singular values and corresponding singular vectors. From these singular vectors, we can produce multiscale embeddings analogous to the scaling functions of diffusion wavelets.

In this paper, we propose a random walk method for learning node embeddings for networks that preserves commute time distances between nodes. Rather than applying the SkipGram model directly to the random walk, we begin by finding a sparse, low-dimensional approximation of the random walk. Specifically, we take inspiration from diffusion wavelets [11] to compute compressed versions of the random walk and its dyadic powers. We then produce commute time-preserving node embeddings from the Green function $G(u, v)$ associated with the sparse approximation. Finally, we use SGD to optimally weight each dimension of the embeddings so that cross-entropy loss is minimized. We show that these embeddings compare favorably to those produced by existing methods such as node2vec, LINE, and GraRep, with classification accuracy measured by F1 Macro scores. Additionally, we show that this truncated SVD algorithm allows for faster dimensional reduction than the original diffusion wavelet algorithm, leading to improved performance and accuracy on larger graphs.

Section 2 of this paper provides some necessary background on random walk and SkipGram-based node embeddings, commute time embeddings and an introduction to random walk sparse approximation techniques. In Section 3, we introduce the proposed new method for learning network embeddings, and discuss its implementation and its relationship to other kernel-based embeddings. In Section 4, we elaborate on the numerical aspects of two possible choices of algorithm (diffusion wavelets and SVD) for computing the sparse approximation of the random walk. Finally, Section 5 provides numerical experiments on several graphs, comparing the proposed method to some existing feature learning methods from the literature.

2.2 Background

Before presenting the implementation details of our method, we will first discuss in more depth the tools used at each stage. Our embeddings are based on the application of SGD to a commute time-preserving embedding of a compressed representation of the random walk on the graph. We will elaborate on these topics below, beginning with SGD, the SkipGram model, and some SkipGram-based embedding methods. Then, we will discuss commute times and their embeddings as introduced in [19], and finally the sparse approximation of the random walk.

2.2.1 The Skipgram algorithm and Skipgram-based embeddings

The SkipGram algorithm optimizes vector embeddings by minimizing the loss function

$$L = \sum_i \sum_{j \in walk(i)} -\log\left(P(j|i)\right).$$

Here, i, j are nodes in the network, $walk(i)$ denotes a random walk trial starting at node i, and $P(j|i)$ denotes the conditional probability that a random walk starting at node i will include node j. In the SkipGram model, these conditional probabilities are modeled using softmax units as follows:

$$P(j|i) = \frac{\exp(v_i \cdot v_j)}{\sum_k \exp(v_i \cdot v_k)},$$

where v_k denotes the vector embedding for node k. For large networks with many nodes, this denominator becomes impractical to compute. To overcome this, *negative sampling* [26] may be used instead. This replaces the softmax units with

$$P(j|i) = \sigma(v_i \cdot v_j) \prod_{m=1}^{M} \sigma(-v_i \cdot v_{k_m}),$$

where $\sigma(x) = 1/(1 + e^{-x})$ and M is the chosen number of negative samples (sampled from a uniform or unigram distribution).

The simplest SkipGram-based method for network embedding is DeepWalk [29]. In DeepWalk, an unbiased random walk is simulated on the graph. Essentially, the walk has an equal chance of stepping to any node which is connected by an edge to the current node, so the transition matrix is a row-normalized adjacency matrix. If the graph is weighted, the weighted adjacency matrix (with entries given by edge weights) may be used instead. Stochastic gradient descent is then used to minimize the above SkipGram objective based on the random walk simulations; that is, at each data point from the random walk trials, the gradient of the loss function with respect to the parameters θ of the embedding function is computed, and the parameters are updated by $\alpha \frac{dL}{d\theta}$ for some learning rate α (see Algorithm 2.1).

Algorithm 2.1 SkipGram Model (stochastic gradient descent).

```
**for** $i$ **do**
  **for** $j \in walk(i)$ **do**
    $L(\theta) = -\log P(v_j(\theta)|i)$
    $\theta = \theta - \alpha \frac{dL}{d\theta}$
  **end for**
**end for**
```

The node2vec [18] method follows a similar outline, but with a slightly more complex choice of random walk. In particular, two parameters p and q are used to bias the random walk simulations. The return parameter p is used to control the probability that a random walk will backtrack to immediately revisit the previous node (e.g., by stepping from node A to B, then immediately back to A). The in-out parameter q controls the likelihood that the walk will step to a node which is close to the previous node. That is, for smaller q, the random walk will be more likely to explore parts of the graph that are further away from the starting node. The resulting random walk's transition probabilities are defined as follows, where the current state of the random walk is n_0 and the previous state was n_{-1}. If n_0 is connected to M other nodes m_1, m_2, ...,m_M, one assigns each m_j an unnormalized probability $\beta(j)$:

$$\beta(j) = \begin{cases} 1/p & \text{if } m_j = n_{-1} \\ 1 & \text{if } m_j \text{ shares an edge with } n_{-1} \\ 1/q & \text{if } m_j \text{ does not share an edge with } n_{-1}. \end{cases}$$

To get the transition probabilities for the next step of the random walk, the $\beta(j)$ are normalized so that they sum to one. Notice that this random walk depends on the two previous steps, and so is not a Markov chain. These parameters can be used to find the ideal balance between breadth-first sampling, which tends to stay nearer to the source node and get a more accurate picture of its immediate neighborhood, and depth-first sampling, which tends to visit parts of the graph further from the source. By finding optimal values of p and q, then, one can engineer a new random walk that will provide superior node embeddings.

In this article, our goal is similar to that of node2vec: we aim to find a choice of random walk that will lead to better node embeddings.

2.2.2 Network Embeddings with Commute times

For a given network Γ, let S denote the set of nodes, E the set of edges, and let A be the adjacency matrix with entries given by the weights a_{ij}. Let D be the diagonal matrix with $D(i,i) = d_i = \sum_j a_{ij}$. The volume of the graph will be given by $vol = \sum_i d_i$, and the *random walk Laplacian* is defined as $L = I - D^{-1}A$. The *normalized Laplacian* has the form $\mathcal{L} = D^{-1/2}LD^{-1/2}$ with eigendecomposition $\mathcal{L} = \Phi\Lambda\Phi^T$, where Λ denotes the diagonal matrix of eigenvalues ordered so that $0 = \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_{|S|}$, and the eigenvectors are given by columns of $\Phi = (\phi_1, \phi_2, ...\phi_{|S|})$. The discrete Green function is the left inverse of the operator L and can be given as

$$G(i,j) = \sum_{k=2}^{|S|} \frac{1}{\lambda_k} \left(\frac{d_j}{d_i}\right)^{1/2} \phi_k(i)\phi_k(j).$$

Here, we assume that the graph is connected. Otherwise $\lambda_k = 0$ for some $k \geq 2$, and we can instead consider the Moore-Penrose pseudoinverse, $L^+ = (L^TL)^{-1}L^T$, to which the methods and results can be easily extended.

We can further define the commute time $CT(i,j)$ to be the mean time for the Markov process on the graph prescribed by L to travel from i to j and back, which also satisfies

$$CT(i,j) = vol\left(\frac{G(i,i)}{d_i} + \frac{G(j,j)}{d_j} - \frac{G(i,j)}{d_i} - \frac{G(j,i)}{d_j}\right). \tag{2.1}$$

It is shown in [19] that the coordinate matrix for embeddings that preserve commute time is then given by $\Theta = \sqrt{vol}\Lambda^{-1/2}\Phi^T D^{-1/2}$, so $\Theta^T\Theta = vol \cdot GD^{-1}$. Thus, commute time embeddings are equivalent to those produced by kernel PCA on a constant multiple of the matrix GD^{-1}.

2.2.3 Computing sparse approximations of Green functions via Schultz method

As we will see, one key component of our method will involve computing an approximate, compressed version of the Green function corresponding to the random walk on the network. Here we use the Schultz method for this purpose. The Schultz method iteratively computes the Green function (or Moore-Pembrose pseudoinverse) associated with L through $G_{k+1} = G_k(2I - LG_k)$. The Green function G can be associated with a diffusion process (i.e., a Markov random walk) defined by the transition matrix $T = I - L = D^{-1}A$. Using the fact that $L = I - T$, and taking $G_0 = I + T$, we obtain:

$$\begin{aligned}
G_1 &= G_0(2I - (I-T)(I+T)) = G_0(2I - I + T^2) = (I+T)(I+T^2),\\
G_2 &= G_1(2I - (I-T)(I+T)(I+T^2)) = G_1(2I - (I-T^2)(I+T^2)\\
&= (I+T)(I+T^2)(I+T^4),\\
&\vdots\\
G_{K+1} &= \prod_{k=0}^{K}(I + T^{2^k}).
\end{aligned}$$

The Green's function G then has the following form on the complement of the eigenspace for the eigenvalue equaling 1, as shown in [11]:

$$Gf = \sum_{k=1}^{\infty} T^k f = \prod_{k=0}^{\infty}(I + T^{2^k})f.$$

Compressed versions of these dyadic powers T^{2^k} can be computed using either the singular value decomposition or diffusion wavelets [11], as we will see in Section 4. Since these matrices are all sparse under certain assumptions, this product can be approximated quickly, which in turn allows for the quick computation of the commute times via the formula (2.1).

Diffusion wavelets [11] extends ideas from wavelet theory to a more general setting. In particular, it extends the descriptions of wavelet bases in [13] for multiscale analysis of networks. The multiscale analysis includes a sequence of subspaces in $L^2(\mathbb{R})$ such that

$$\cdots V_{-1} \supset V_0 \supset V_1 \supset V_2 \supset \cdots,$$

$\bigcap_m V_m = \emptyset$, and $\overline{\bigcup_m V_m} = L^2(\mathbb{R})$. The V_m also have the property that $f \in V_m \Leftrightarrow f(2\cdot) \in V_{m-1}$. Thus larger m correspond to increasingly coarser scales.

Scaling functions are constructed from some $\Phi \in V_0$ by dilating and scaling according to $\Phi_{m,n}(x) = 2^{-m/2}\Phi(2^{-m}x - n)$. The scaling functions at scale m, $\{\Phi_{m,n} : n \in \mathbb{Z}\}$, form an orthonormal basis of the space V_m.

[13] also shows that a second basis of orthonormal functions Ψ_m can be constructed for each scale m. The general construction is as follows: given the scaling function Φ, choose wavelet coefficients $c_n, n \in \mathbb{Z}$ such that

$$\Phi(x) = \sum_n c_n \Phi(2x - n).$$

Then, we choose

$$\Psi(x) = \sum_n (-1)^n c_{n+1} \Phi(2x + n).$$

We can then define wavelet bases at each scale m from dilations and translations of the function Ψ, similarly to the scaling functions (that is, $\Psi_{m,n}(x) = 2^{-m/2}\Psi(2^{-m}x - n)$). The $\Psi_{m,n}$ are orthonormal and span a space W_m with the properties $W_m \perp V_m$ and $W_m \bigoplus V_m = V_{m-1}$. Furthermore, the entire collection of $\Psi_{m,n} \forall m, n \in \mathbb{Z}$ is an orthonormal wavelet basis for $L^2(\mathbb{R})$.

In the diffusion wavelets setting [11], the multiscale analysis includes a sequence of subspaces

$$V_0 \supset V_1 \supset V_2 \supset \cdots$$

such that V_0 is the range of the operator T, and for $m \in \mathbb{Z}$, V_m represents an approximation of the range of T at scale m. In particular, when $\{\zeta_i\}_{i \in \mathbb{Z}^+}$ and $\{\lambda_i\}_{i \in \mathbb{Z}^+}$ denote respectively the eigenvectors and eigenvalues of T, V_m is defined as the span of $\{\zeta_\lambda : \lambda^{2^{m+1}-1} \geq \varepsilon\}$. Each W_m can then be defined as the orthogonal complement of V_m in V_{m-1}, as in the original wavelet setting. Note that the parts of the spectrum with $\lambda^{2^{m+1}-1} \geq \epsilon$ can be viewed as "low-pass," so that the V_m correspond to the "approximation" or "smoothing" subspaces in classical wavelet theory, and the W_m correspond to the "detail" components. The scaling functions Φ_j at each level j then provide a basis for V_j, approximating a dyadic power of T. We leave the numerical details of the algorithm for Section 4.

2.3 An implementation of the SkipGram model based on sparse approximation and commute times

We now propose a method combining ideas from the previous section to produce network embeddings that preserve commute times while also optimizing the cross entropy loss. Overall, the idea is to first obtain a sparse approximation of the random walk from a multiscale embedding process, and produce a commute time-preserving embedding basis from this approximation. Then, the final embeddings are produced via SGD, optimizing the weights on these basis vectors to minimize the cross entropy loss.

Let T denote the transition matrix for the random walk on a network with N nodes.. We here assume that T is local in the sense that its columns have small support, and that high powers of T will be of low rank. We start by computing the compressed representations of the dyadic powers of T, and call these representations $T_1, \ldots, T_k$, so that T_k denotes the representation of T^{2^k}. To do this, we apply an orthogonalization procedure (the details of which will be explained in Section 4) to the columns of T, producing an orthogonal basis $U_1 = \{u_1, u_2, \ldots u_K\}$, $\{K \leq N\}$, that approximates the range of T. Then, we can write $T_1 := U_1^T T U_1$, so that T_1 is an approximation of T^2 expressed in terms of the basis U_1. We then repeat the process, orthogonalizing the columns of T_1 to produce an approximate basis U_2 for its range, using this new basis to compute a representation of $T_2 \approx T^4$, and so on.

More specifically, at each step we compute a truncated SVD of T_k. The singular vectors associated with the largest singular values then form a basis spanning approximately the range of T_k, which we can view as a ”low-pass" approximation subspace analogous to the low-pass subspaces of classical wavelet theory. We can then use these singular vectors to compute a compressed representation of the next dyadic power, $T_{k+1} \approx T^{2^k}$, with respect to the basis U_k. Further details of this process will be reserved for Section 4.

Using the Schultz method described in Section 2.3, we compute the unnormalized Green function G from $T_1, \dots, T_k$. From [19], network embeddings that preserve commute time relationships have the embedding matrix Θ satisfying $\Theta^T\Theta = vol \cdot GD^{-1}$. We produce such multiscale embeddings by repeating the same approximation process to produce compressed representations of $Q = vol \cdot GD^{-1}$.

This process yields at each scale a basis of scaling functions $\tilde{U}_k$ that spans an approximation of the range of $Q_k \approx Q^{2^k}$, as described in Section 4. Then, the embeddings for compression level k (with $k = 1$ representing the finest scale, and larger k representing increasingly coarse scales) are constructed from the scaling functions of Q_k. Essentially, to obtain the k-level embedding for node x, we extend the truncated singular value decomposition at the kth level, $Q_k \approx \tilde{U}_k\tilde{\Sigma}_k\tilde{V}_k^T$, to the original basis and take the matrix $\Theta = \tilde{\Sigma}_k^{1/2}\tilde{U}_k^T$ to be the embedding matrix; that is, the nth column of Θ will provide the embedding coordinates for the nth node. (For symmetric G, note that $vol \cdot GD^{-1} \approx \Theta^T\Theta$, as desired.)

The singular vectors are a sensible choice for the embeddings because, similarly to the scaling functions in classical wavelet theory, the span of the columns of U_k is a low-pass approximation space for the range of Q_k. Similar multiscale embeddings are also utilized in, for example, [11], [44], although these embeddings utilize the scaling functions of diffusion wavelets rather than singular vectors.

It is also possible to replace the truncated SVD algorithm used here with the diffusion wavelet algorithm of [11]. A more detailed explanation and comparison of the two algorithms is given in Section 4. In the numerical experiments performed here, the SVD version of this algorithm outperformed the diffusion wavelet algorithm, so the remainder of the paper will focus on the SVD version. The same ideas can be implemented for either version.

While the matrix Θ can itself be used as an effective node embedding, we can further improve our results by using this sparse approximation as a starting point to obtain node embeddings that minimize cross entropy. The commute-time-based embedding results in a basis spanning the range of (a power of) the sparse approximation to the random walk. We can reweight the vectors in this basis spanning the same space such that cross entropy loss is minimized. Additionally, we will with some probability reintroduce singular vectors that were removed in the previous truncations. In this way, we randomly reintroduce some of the information that has been lost in the approximation steps, without compromising computational efficiency. In particular, since taking powers of a Markov matrix is equivalent to running the Markov process forward in time, reintroducing columns from previous steps is analogous to preserving information about different time scales.

Cross-entropy loss is given by the following:

$$Loss_{CE} = \sum_{i} \sum_{j:(i,j)\in E} -\log(P(j|i)), \tag{2.2}$$

where $P(j|i)$ denotes the conditional probability of node j conditioned on i (the probability that a random walk will move from node i to node j). Here we model the conditional probabilities using the angular distances between nodes,

$$AD(i,j) = 1 - \frac{1}{\pi} \arccos\left(\frac{\theta_i^T \theta_j}{||\theta_i||||\theta_j||}\right),$$

by taking $P(i|j) = \frac{1}{N} AD(i,j)$ if $i \neq j$ and $P(j|j) = 1 - \sum_{i\neq j} P(i|j)$.

Given the $N \times K$ sparse approximation Θ, where the ith row of Θ, θ_i, provides an embedding of node i, we aim to find constants $C_1, C_2, ...C_K$ such that the embedding given by ΘC where $C = diag(C_1, C_2, ...C_K)$ minimizes cross entropy (2.2). Each basis vector, given by the columns of Θ, is therefore multiplied by a different weight constant. Using SGD, we choose the optimal constants so that the role of the more "important" columns (which correspond to particular singular values of the approximation to the random walk) will be emphasized.

After each step of SGD, with some small probability δ, we also extend the basis θ with one of the indices from the basis in the previous step. That is, if $\Theta = \tilde{\Sigma}_k^{1/2} \tilde{U}_k^T$, we randomly select an index $\alpha^k N \leq j \leq \alpha^{k-1} N$. This then corresponds to a column of U_{k-1}, u_j. We can then project $\sigma_j u_j$ onto the basis U_k, and append this projection to Θ. This allows for the reintroduction of some information from the ”high-pass" subspaces, removed during the approximation process, while still benefiting from the low-dimensional compression given by the low-pass bases.

The overall algorithm is as follows:

1. Produce the sparse approximation of the random walk (e.g. the Green function G and the commute time embeddings θ) as described above, by computing the first $\alpha^k N$ singular values and left singular vectors at the kth step.

2. Initialize C as the identity matrix, normalized so that its diagonal entries sum to 1.

3. Via stochastic gradient descent, find C optimizing the cross entropy loss function (2.2).

4. During the SGD process, at each step (with probability δ) append to the embedding matrix a vector corresponding to an index which was previously removed in the truncated SVD. Also extend C accordingly.

As a visualization aid, we consider the Zachary's Karate Club network [50]. The scatter plot in Figure 2.1 (b) shows the two-dimensional embeddings generated by Commute Times with Diffusion Wavelets for the karate club network. The thirty-four nodes of this graph each represent a member of the karate club, with edges representing whether or not two people communicate outside of the club. Each node has been assigned a group label (indicated by the color of that node in the plot) corresponding to one of four clusters, depicted in Figure 2.1 (a). The above method results in embeddings that successfully distinguish all four clusters.

Table 2.1 Performance and clustering score for the karate club graph embeddings in Figure 2.3.

Network	d	**Runtime**	**F1 Macro**
Karate Club	3	0.85 s	0.9177

2.3.1 Modified Johnson-Lindenstrauss Lemma

It can be shown that the embeddings produced via the proposed method (i.e., Green function-based commute time methods) satisfy a modified version of the Johnson-Lindenstrauss lemma. In

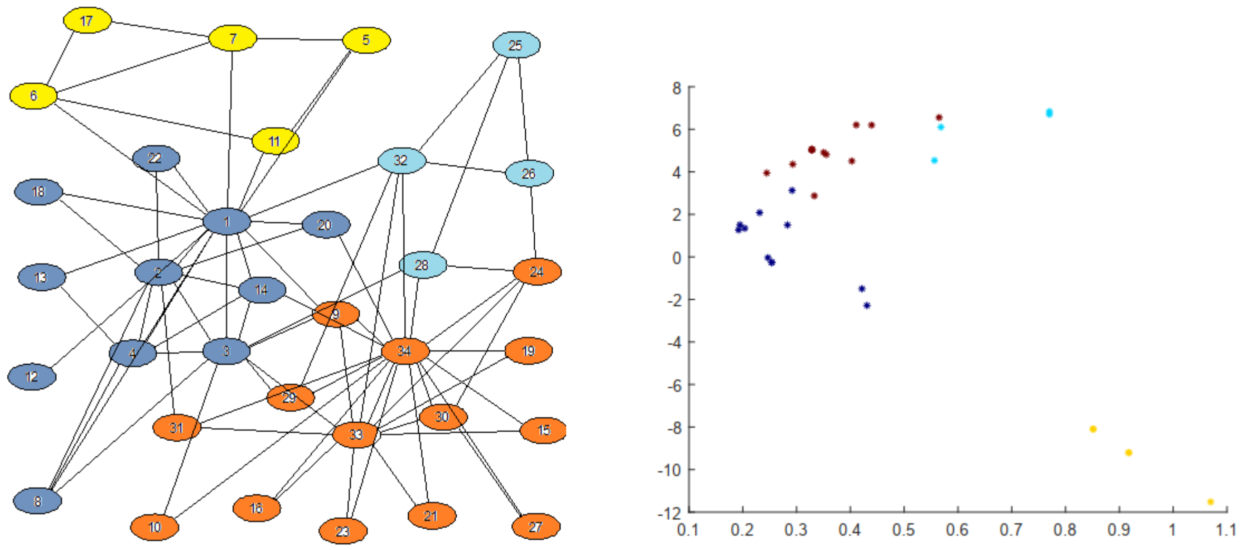
Figure 2.1 a) The karate club network (created via [30]). b) Commute time embeddings found via the proposed method. Colors of points are determined by the true classification of each node.

particular, the Euclidean distances between these embeddings approximate up to ε the commute time distance (or, for diffusion wavelet-based embeddings, the square root of the commute time distance) between the original nodes.

The commute time can be approximated with high probability by the product of $G^{1/2}$ and a constant multiple of a random matrix with normally distributed entries:

Theorem 1. *Let S be the set of all vertices in a network with unnormalized Green function given by G, degree matrix D, and volume vol, and let $CT(\cdot,\cdot)$ denote the commute time. There exists $\Pi \in \mathbb{C}^{m\times d}$ where $m = O(\log(1/\delta)/\varepsilon^2)$ and $\varepsilon > 0$ such that for all $x, y \in S$,*

$$(1-\varepsilon)CT(x,y) \leq ||\Pi(volGD^{-1})^{1/2}\delta_x - \Pi(volGD^{-1})^{1/2}\delta_y||_2^2 \leq (1+\varepsilon)CT(x,y) \tag{2.3}$$

with probability $1-\delta$.

Next, we give the main result of this section, which demonstrates that our embeddings preserve commute times between nodes.

Theorem 2. *Let S be the set of all vertices in the network, and let $CT(\cdot,\cdot)$ denote the commute time.*

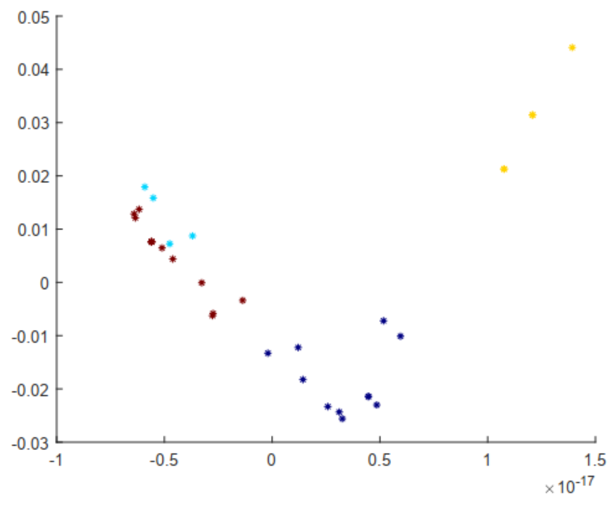
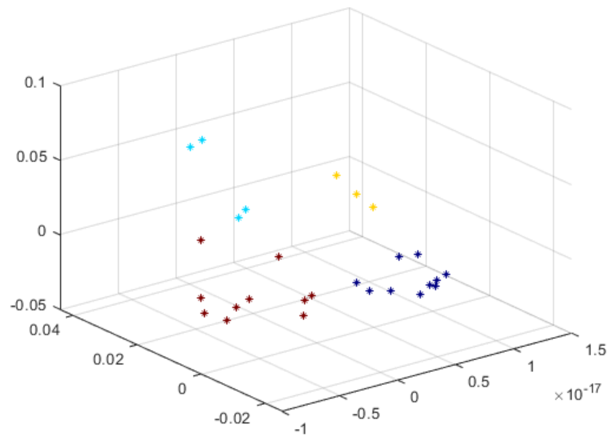

Figure 2.2 2 and 3-dimensional embeddings of the karate club network, based on the first columns of the commute time embedding matrix from [19].

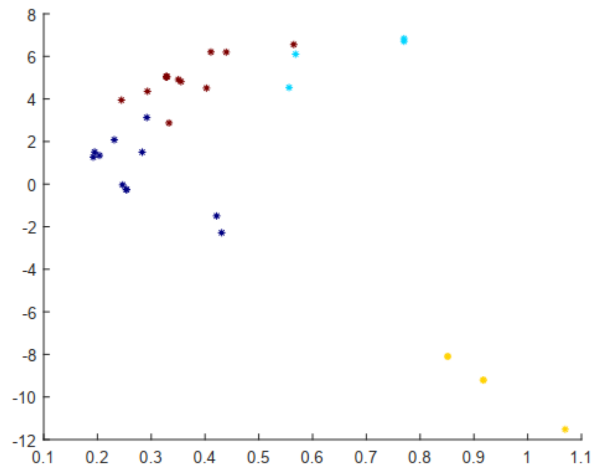
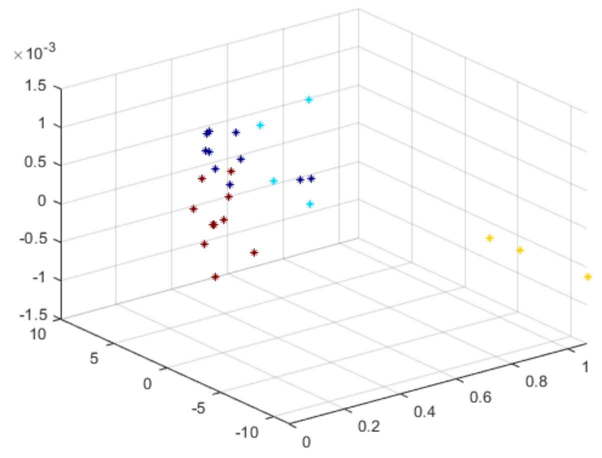

Figure 2.3 2 and 3-dimensional embeddings of the karate club network, created with the proposed method.

1. *If $\Phi(x) := \Phi^T\delta_x$ denotes the embedding for node x obtained by applying the diffusion wavelet algorithm to the matrix $volGD^{-1}$ with precision constant ε (that is, Φ is the orthonormal basis of scaling functions and $\Phi(x)$ is the xth row of Φ), then for all $x, y \in S$,*

$$-\varepsilon + \sqrt{CT(x,y)} \leq ||\Phi^T(\delta_x - \delta_y)|| \leq \varepsilon + \sqrt{CT(x,y)}. \tag{2.4}$$

2. *Let $U\Sigma V^T \approx volGD^{-1}$ represent the truncated SVD using the αN leading singular values $\sigma_1 \geq ... \geq \sigma_{\alpha N}$, for some $\alpha \leq 1$. Let Θ denote the embedding matrix obtained from the αN leading left singular vectors U, such that $\Theta = (\Sigma^{1/2})^T U^T$. Then for all $x, y \in S$, and $\varepsilon = O(\sigma_{\alpha N+1})$,*

$$-\varepsilon + CT(x,y) \leq ||\Theta^T(\delta_x - \delta_y)|| \leq \varepsilon + CT(x,y). \tag{2.5}$$

See the Appendix for proofs.

2.3.2 Relationship with diffusion maps

We here show that embeddings given by diffusion wavelet scaling functions are equivalent up to a rotation to the embeddings produced from a given kernel eigenmap method. In particular, the embeddings produced via the application of diffusion wavelets to the matrix $vol \cdot GD^{-1}$ are equivalent up to a rotation to the embeddings given by kernel PCA on $vol \cdot GD^{-1}$, and so they are also equivalent up to a rotation to the commute time embeddings from [19].

In [10], it is shown that any kernel eigenmap method can be viewed as solving

$$\min_{Q_2(f)=1} Q_1(f), \text{ where } Q_1(f) = \sum_x Q_x(f),$$

where Q_2, Q_x are symmetric positive semi-definite quadratic forms with Q_x local. For example, for a kernel k, we can take

$$Q_1(f) = \sum_x \sum_y k(x,y)(f(x) - f(y))^2$$
$$Q_2(f) = \sum_x v(x) f(x)^2.$$

Then, $Q_2^{-1}Q_1$ represents a discretization of a differential operator (i.e., a Laplacian).

The following theorem and its proof are a slight generalization of the proof of Theorem 1 from [44] (see Appendix for proof).

Theorem 3. *Suppose we have a kernel-based method (e.g. Q_1, Q_2 satisfying the properties in diffusion maps). Then the embeddings obtained from the diffusion wavelet scaling functions for*

$Q_2^{-1}Q_1$ at level j are equivalent, up to a rotation, to the p_j-dimensional embeddings obtained from the diffusion maps embedding, where p_j gives the number of scaling functions at scale j.

In particular, since the commute time embeddings in [19] are equivalent to kernel PCA on the matrix $vol \cdot GD^{-1}$, we can apply the above theorem with $Q_2^{-1}Q_1 = vol \cdot GD^{-1}$ to see that the embeddings produced via the diffusion wavelet method are equivalent up to a rotation to the p_j-dimensional commute time embeddings, and thus these embeddings will also preserve commute times between nodes.

2.4 Numerical Details of Sparse Approximation

We now elaborate on two possible algorithms for the sparse approximation of the random walk. We first give details of the diffusion wavelet algorithm introduced in [11], and then propose an alternate implementation using an approximate SVD.

2.4.1 Diffusion Wavelets

Diffusion wavelets generalizes the wavelet theory of [13] to diffusionlike operators. We can apply this theory for random walks on networks by taking the transition matrix T for a Markov process on the network to be such an operator. In this setting, the scaling functions Φ_m at level m are constructed by orthonormalizing the columns of the representation of T at the previous level, T_{m-1}, up to precision ε–that is, the vectors Φ_m ε-span the range of T_{m-1}, defined as follows in [11].

Definition. Let $\mathcal{H}$ be a Hilbert space such that $\{w_j\}_{j\in J} \subset \mathcal{H}$. A set of vectors $\{v_i\}_{i\in I}$ **ε-spans** the set of vectors $\{w_j\}_{j\in J}$ if

$$||P_{\langle\{v_i\}_{i\in I}\rangle} w_j - w_j||_{\mathcal{H}} \leq \varepsilon,$$

where $P_{\langle\{v_i\}_{i\in I}\rangle}$ represents orthogonal projection onto the span of $\{v_i\}_{i\in I}$. We also say that $\langle\{v_i\}_{i\in I}\rangle$ is an ε-span of $\langle\{w_j\}_{j\in J}\rangle$.

In fact, the subspace spanned by these scaling functions is an ε-approximation of V_m, which we will denote $\tilde{V}_m$. As we will see, T_m is a representation of T_{m-1}^2 by construction, so each step of the diffusion wavelets algorithm dilates the scale by a factor of 2, as in the original wavelet setting.

In classical wavelet theory, the localization of wavelet bases can better approximate choppy signals and signals with different time scales compared to Fourier series, since using Fourier series in these cases would require one to break up the domain into pieces and analyze each piece separately. Using the local diffusion wavelet scaling function bases provides a similar advantage over global eigenvalues, particularly for graphs with irregular structures.

In the following overview of the diffusion wavelet algorithm, we borrow some notation from [11]: we will use $[A]_{B_1}^{B_2}$ to denote a matrix A that is written with respect to the basis B_1 in the domain, and with respect to the basis B_2 in the range.

For an operator T such that the numerical rank of its powers $\{T^k\}_{k=0,1,2,...}$ decreases with k, the diffusion wavelets algorithm [11] computes compressed representations of the dyadic powers T^{2^k}, which can then be used to efficiently compute certain functions of T, in particular the Green function. In addition, when T is an $|S| \times |S|$ Markov matrix, taking higher powers of T is equivalent to running the Markov chain forward in time. Thus, diffusion wavelets allows for the analysis of the random walk at different time scales.

The diffusion wavelets algorithm $\mathcal{DW}(T, K, \epsilon)$ is outlined as follows: As inputs, $\mathcal{DW}$ takes the operator $T = T_0$, a parameter K determining the number of iterations to be performed, and a precision parameter ϵ. The columns of T are originally expressed in the basis $\{\delta_i\}_{1 \leq i \leq |S|} =: \Phi_0$.

First, the columns of T are orthonormalized up to ϵ using a modified QR algorithm (such as modified Gram-Schmidt with pivoting or rank-revealing QR; [11] elaborates on possible choices of orthogonalization technique). This results in an orthonormal matrix that gives an ε-span Φ_1 of the columns of T written in terms of the original basis Φ_0, which we here denote $[\Phi_1]_{\Phi_0}$. Φ_1 forms a basis for the subspace $\tilde{V_1}$, and gives an ϵ-approximation of V_1 (that is, the range of T). Additionally, the rank-revealing QR algorithm guarantees that the elements of this basis are well localized. We also obtain from the QR decomposition an upper triangular matrix representation of T, expressed with respect to Φ_0 in the domain and Φ_1 in the range, denoted $[T]_{\Phi_0}^{\Phi_1}$.

Then, the product $[\Phi_1]_{\Phi_0}[T^2]_{\Phi_0}^{\Phi_0}[\Phi_1]_{\Phi_0}^{*} = [T^2]_{\Phi_1}^{\Phi_1} =: T_1$ provides a representation of T_0^2 in terms of Φ_1. It is also possible, if desired, to compute a basis of wavelet functions spanning the orthogonal complement of $\tilde{V}_1$ in $\tilde{V}_0$, denoted $\tilde{W}_1$, by orthonormalizing the columns of $I - [\Phi_1]_{\Phi_0}[\Phi_1]^*_{\Phi_0}$. By iterating this entire process, we can compute representations of $[T^{2^k}]^{\Phi_{k-1}}_{\Phi_{k-1}} =: T_k$, scaling functions $[\Phi_k]_{\Phi_{k-1}}$ spanning the space $\tilde{V}_k$ that ε-approximates V_k (i.e. the range of T_k), and wavelet functions Ψ_k for levels $k = 1, 2, ...$

2.4.2 SVD

In the numerical experiments presented here, we found that an SVD-based multiscale analysis was in general faster and produced more accurate node classifications compared to the original diffusion wavelet algorithm. This is demonstrated in Table 2.2 and Figure 2.4; on the butterfly dataset, three-dimensional embeddings produced via the SVD algorithm perform better than either the three-dimensional diffusion wavelet embeddings or the first three columns of the commute time embeddings described in [19]. Additionally, the SVD approach allows more control over the rate of dimensional reduction, since one can directly choose the proportion of singular values to be retained. This alternative formulation is as follows:

Table 2.2 Comparisons between three-dimensional diffusion wavelet-based embeddings, truncated SVD-based embeddings, and the commute time embeddings introduced in [19], on the butterfly dataset [43, 45]. Runtime represents the time required to compute sparse approximations for powers of T and G via the given algorithm.

Method	**Runtime**	**F1 Macro**
Diffusion Wavelets	70 s	0.5827
SVD	3.5 s	0.7490
Commute Time Embeddings	0.5 s	0.6051

First, we compute a truncated SVD of T, retaining the j largest singular values and the corresponding singular vectors, so that $T \approx U\Sigma V$ where $U \in \mathbb{R}^{|S|\times j}$, $\Sigma \in \mathbb{R}^{j\times j}$, and $V \in \mathbb{R}^{j\times|S|}$. Then, $U_1 := U$ provides an orthogonal basis for approximately the range of T, for some margin of error determined by the choice of j. To obtain a representation of T^2 with respect to the basis U_1, we use $T_1 := U_1^T T U_1$. To justify this, note that the left singular matrix U has orthogonal columns, and furthermore these columns span the range of T. In particular, the left singular vectors corresponding to the largest singular values span an approximation of the range of T, and therefore they play a role analogous to the diffusion wavelet scaling functions.

To obtain representations for T^{2^k} for $k = 2, 3, ...$, we can simply iterate this process by finding the truncated SVD of T_{k-1}, taking U_k to be the left singular matrix of that decomposition, and setting $T_k = U_k^T T_{k-1} U_k$. Additionally, each U_k can be represented in the original basis using the change-of-basis formula:

$$[U_j]_{U_0} = [U_{j-1}]_{U_0} [U_j]_{U_{j-1}} = \cdots = [U_0]_{U_0} [U_1]_{U_0} \cdots [U_{j-1}]_{U_{j-2}} [U_j]_{U_{j-1}},$$

where $[U_j]_{U_k}$ represents U_j expressed in terms of the columns of U_k, and U_0 represents the original basis of the columns of T. (Note that in this notation, $[U_j]_{U_{j-1}} = U_j$.)

In the case where the random walk matrix T is symmetric, the connection to the diffusion wavelet-based algorithm is obvious; the singular values are the absolute values of the eigenvalues of T, and therefore the largest singular values are the largest-magnitude eigenvalues. The columns of the left singular matrix U then give the eigenvectors of T. If we select j such that $\forall k \leq j$, $\lambda_k^{2^{m+1}-1} \geq \varepsilon$, then the first j columns of U (when the singular values are ordered from largest to smallest) must span $V_m := \langle\{\zeta_\lambda : \lambda^{2^{m+1}-1} \geq \epsilon\}\rangle$.

In the asymmetric case, the basis given by the first j columns of U approximates the range of T according to the largest singular values, rather than the eigenvalues. Here, the multiscale analysis involves a sequence of subspaces

$$X_0 \supset X_1 \supset X_2 \supset \cdots$$

such that X_0 is the range of the operator T, and for $m \in \mathbb{Z}$, X_m represents an approximation of the range of T at scale m, obtained via the largest singular values and their corresponding singular vectors. In particular, when $\{\sigma_i\}_{i\in\mathbb{Z}^+}$ denotes the singular values of T, and u_{σ_i} denotes the singular vector corresponding to the ith singular value, we can define $X_k := \langle\{u_\sigma : \sigma^{2^{k+1}-1} \geq \varepsilon\}\rangle$. These subspaces X_k can, as in the diffusion wavelet construction, be viewed as the "low-pass" portions of the range of T, corresponding to the "approximation" or "smoothing" subspaces in classical wavelet theory.

Algorithm 2.2 Truncated SVD: An approximate singular value decomposition of the $N \times N$ matrix T using the largest αN singular values.

```
**Input:** Matrix $T \in \mathbb{R}^{N\times N}$, fraction of singular values retained $\alpha \in (0, 1)$
**Result:** $\hat{U}, \hat{\Sigma}, \hat{V^T}$ = TruncSVD($T$, $\alpha$)

$U, \Sigma, V^T = SVD(T_{k-1})$ ;      /* Singular values sorted in decreasing order */

$\hat{U}$ = first $\alpha N$ columns of $U$
$\hat{\Sigma}$ = first $\alpha N$ columns of $\Sigma$
$\hat{V}^T$ = first $\alpha N$ rows of $V^T$.
```

Algorithm 2.3 Commute Time-based Network Embeddings with Truncated SVD.

```
**Input:** $T$, $K$, $K_2$, $0 \leq \alpha \leq 1$
**Result:** Node embeddings for the network represented by the $|S| \times |S|$ Markov matrix $T$

$T_0 = T$
**for** $1 \leq k \leq K$ **do**
    $U, \Sigma, V$ =TruncSVD$(T_{k-1}, \alpha)$
    $T_k = U_k^T T_{k-1} U_k$
**end for**
$G = I + T_0$
$G = G \prod_{k=1}^{K_1} U_k(I + T_k)$ ;                                /* Compute $G$ */

**for** $1 \leq k \leq K_2$ **do**
    $\tilde{U}, \tilde{\Sigma}, \tilde{V}$ =TruncSVD$(vol \cdot G \cdot D^{-1}, \alpha)$
    $\theta_k^T = \tilde{U}_k \tilde{\Sigma}_k^{1/2}$ ;              /* Rows of $\theta_k^T$ give embeddings at level $k$ */

**end for**
Embedding Matrix= $\theta_{K_2}$.
```

One possible disadvantage of this SVD method is that the orthogonalization here is not as careful as the orthogonalization step in diffusion wavelets [11], and as a result our bases are not guaranteed to be localized. While this has no apparent adverse effects on the applications discussed here, this method may not be suitable for other situations where diffusion wavelets is appropriate, particularly situations where the localization of the scaling functions is crucial. However, in practice (at least for the examples discussed here), the SVD basis vectors are "approximately" localized– that is, for any given basis vector, the majority of the nodes will be clustered near 0. Setting a tolerance such that these low-magnitude values is replaced with zero may improve performance slightly, in such cases.

This "approximate localization" property can be seen in Figure 2.4, which shows embeddings based on 2 and 3 singular vectors. Because this is smaller than the intrinsic dimension of the dataset, most of the nodes take values near 0 on the first 3 singular vectors.

2.5 Examples and performance comparisons

The following examples illustrate the effectiveness of the method for multi-label classification tasks. In this setting, each node is assigned a class label. The goal is to correctly predict the nodes' class labels based on their embeddings. In the following experiments, we record for each trial the computational time, and compare the F1 macro scores to judge the relative quality of each set of label predictions.

We compare our method to the following existing methods:

- GraRep [6], which creates multiscale embeddings by using the SVD of the k-step transition probability matrix to obtain k-step representations, and then concatenating the representations from each scale. Here we consider both a Matlab implementation [6] and a Python version available from [34].

- DeepWalk [29], which uses SkipGram to generate random walk-based embeddings from unbiased random walks. Here we use the code from the original paper, which uses hierarchical softmax to speed up computation of the conditional probabilities.

- Node2vec [18], a method similar to DeepWalk that uses biased random walks to compute embeddings, based on parameters for breadth-first and depth-first sampling.

- LINE [39], which uses an edge-sampling algorithm to optimize an objective function that is designed to preserve both first and second-order relationships.

The Commute Time with SVD embeddings were computed using Algorithm 2.3 from Section 3. The choice of embedding size d for the commute time-based embeddings varies for each dataset, depending on both the size of the original network and the number of iterations of the compression algorithms used. For GraRep, LINE, and DeepWalk, all parameters except embedding size d

were left at their default values, and the sizes of the embeddings were chosen to match those of the Commute Time with Diffusion Wavelet embeddings. The F1 Macro scores reported here are averages based on 10 trials. All computations were performed with an Intel (R) HD Graphics 520 GPU.

The F1 macro scores were computed by generating embeddings based on the data, then using the embeddings to train a 5-nearest neighbor classification routine. This classification model was then used to predict a class label for each node based on its embedding. F1 scores for each class were computed using the formula:

$$F1_{\text{class } i} = \frac{TP}{TP + \frac{1}{2}(FP + FN)},$$

where TP denotes the number of true positives (nodes which were correctly assigned to class i), FP denotes the number of false positives (nodes which were correctly assigned to a different class), and FN denotes the number of false negatives (nodes which should have been assigned to class i, but were not). The overall F1 macro score is the sum over i of all $F1_{\text{class } i}$.

For every dataset, the embeddings created with the method presented here obtained the highest or second-highest F1 macro score. The runtime for this method was below or on par with that of methods with comparable F1 scores, with the exception of the butterfly dataset. The small size of this dataset allowed the deterministic GraRep method to run efficiently and produce accurate embeddings. However, on the larger datasets, our method produced higher quality label predictions (as measured by F1 scores) compared to GraRep.

2.5.1 Butterfly dataset

We first consider a network from [43, 45] consisting of 832 nodes which each represent different butterflies. Each of the organisms belongs to one of 10 species, and between 55 and 100 pictures of each species were collected from Google Images [43]. Edges indicate similarity between organisms; that is, two nodes will be connected by an edge if their respective images share features (such as wing color and pattern) in common. The data is available from [21]. The commute time with SVD embeddings were based on 5 iterations of the algorithm, where 50% of the singular values were retained at each iteration.

The embeddings that led to the best F1 macro scores for the butterfly network were Commute Times with SVD and GraRep (see Table 1). The computation time and F1 score of the Commute Time with SVD method were comparable to those of the Matlab and Python GraRep implementations.

Table 2.3 Butterfly dataset.

Method	d	**Runtime**	**F1 Macro**
Commute Times with SVD	27	35 s	0.9212
DeepWalk	27	20 s	0.8692
GraRep (Python)	27	10 s	0.9213
GraRep (Matlab)	27	7 s	0.9103
node2vec (p=0.5, q=2.0)	27	3 min 3 s	0.8739
LINE	27	3.5 min	0.7662

Figure 2.4 plots the three-dimensional and two-dimensional embeddings of this network, respectively. The coordinates of the embeddings were determined by the first columns of the embedding matrix obtained above. Each color represents the correct classification of that node, that is, the butterfly species to which the corresponding image belongs. Although omitting the rest of the dimensions has decreased the quality of the embeddings (the F1 macro score for the three-dimensional embeddings pictured here are only 0.6287 and 0.7490), the clusters corresponding to three of the species are visible. Also note that, for these low-dimensional embeddings, the localized nature of the embedding vectors has caused the majority of the nodes to be clustered around the origin. Slightly higher-dimension embeddings (e.g. $d = 27$, as in Table 2.3) result in more accurate embeddings, as the majority of nodes will take a larger-magnitude value on at least one embedding vector.

2.5.2 Cora dataset

The Cora dataset [36] (available from the Deep Graph Library, [46]) is a citation network consisting of 2708 papers, with edges representing citation relationships. For a given edge, the paper represented

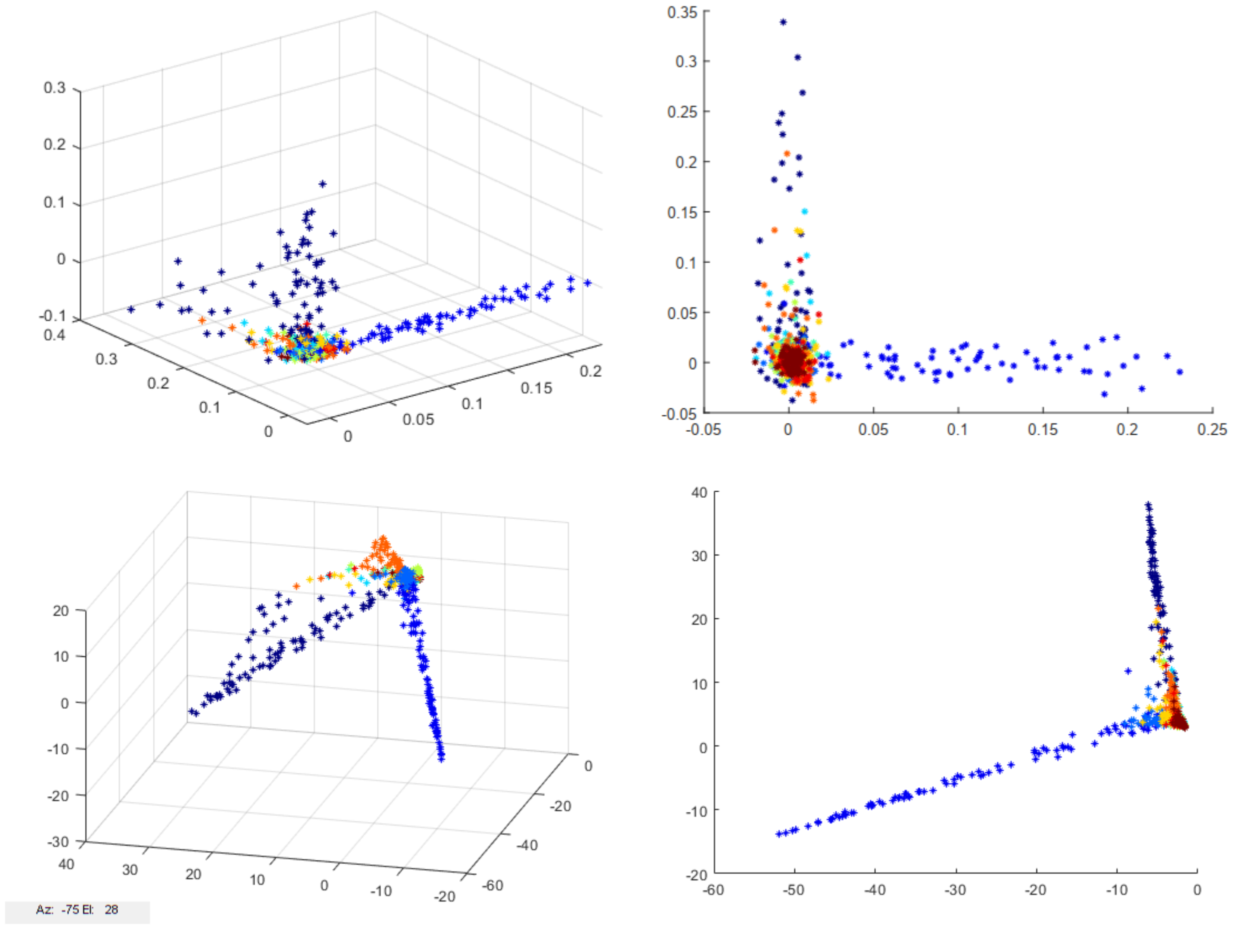
Figure 2.4 Three-dimensional and two-dimensional node embeddings for the butterfly species network, generated using the algorithm described in Section 3. Embeddings in the top row use sparse approximations produced via diffusion wavelets, and those in the bottom row use sparse approximations produced via truncated SVD. Each color represents the species to which that node belongs.

by the source node contained a citation of the target node's paper. The papers are sorted into seven different topics, corresponding to seven distinct classes (Case Based, Genetic Algorithms, Neural Networks, Probabilistic Methods, Reinforcement Learning, Rule Learning, and Theory).

The embedding size of the Commute Time with SVD embeddings is based on 4 iterations of this algorithm, retaining 50% of the singular values at each iteration.

As shown in Table 2, for this network, the highest F1 score was achieved by LINE, with node2vec and the SVD-based commute time methods scoring only slightly lower. However, here the SGD-based methods and LINE had the longest runtimes, while GraRep required less than one minute and still maintained F1 scores above 0.8. Overall, the commute time with SVD method, LINE and node2vec achieved the highest accuracy at the classification task while computing the embeddings efficiently.

Table 2.4 Cora dataset.

Method	d	**Runtime**	**F1 Macro**
Commute Times with SVD ($\alpha = 0.1$)	170	3 min	0.8787
DeepWalk	170	81 s	0.8061
GraRep (Python)	170	37 s	0.8072
GraRep (Matlab)	170	65 s	0.8333
node2vec (p=0.5, q=2.0)	170	4 min	0.8749
LINE	170	2.5 min	0.8864

2.5.3 Amazon co-purchase photo dataset

We next apply the methods to the Photo segment of the Amazon Co-Purchase network [22], available from [46]. The network consists of 7650 products for sale, each of which is assigned a label representing one of eight distinct product categories. Edge relationships are determined by which products were recommended when viewing the product page for a given node. As discussed in [22], these edges can therefore imply a variety of relationships: for instance, that customers frequently view the product pages for both items, or that customers frequently purchase both products, either simultaneously or one after the other. Depending on which type of relationship a given edge refers to, then, two connected nodes might be products that serve the same purpose, two different types of the same product, or products that can be used together. For the commute time with SVD embeddings, the embedding size is based on 4 iterations of this algorithm, where 50% of the singular values are retained at each iteration. The average F1 macro score over 10 trials was 0.9383.

Results for each of the methods are outlined in Table 2.5. The Amazon Co-Purchase network is the largest of the networks considered here, leading to longer computational times for all methods. In these trials, the commute time with SVD method and node2vec had the best performance in terms of accurate labeling, with F1 scores of 0.9383 and 0.9335 respectively, but the commute time with SVD method required approximately 20 minutes to run. In comparison, node2vec also had an F1 score above 0.9, but only required around 10 minutes. The Python implementation of GraRep performed similarly to the SVD-based method, while the Matlab version of GraRep failed to accurately classify the nodes. This is likely due to the structure of the network; specifically, the fact that the degree matrix of this network, which is inverted in the GraRep algorithm, is singular to machine precision.

Table 2.5 Amazon Co-Purchase Dataset.

Method	d	**Runtime**	**F1 Macro**
Commute Times with SVD ($\alpha = 1$)	480	1189 s	0.9383
DeepWalk	480	570 s	0.9167
GraRep (Python)	480	1549 s	0.8972
GraRep (Matlab)	480	920 s	0.0451
node2vec (p=0.5, q=2.0)	480	9.5 min	0.9335
LINE	480	178 s	0.8203

2.5.4 Email database dataset

Finally, we consider a network of email communications from a particular organization [49, 20], available from [21]. This graph contains 1005 nodes, each corresponding to a member of a certain European research institution. Edges indicate that a particular pair of members had exchanged emails during the 18-month data collection time period. Specifically, an edge from X to Y exists only if at least one email was sent from X to Y, and an email was sent from Y to X. Each person in the network belongs to one of 41 departments. The commute time with SVD embeddings were based on 5 iterations of the truncated SVD algorithm, where 75% of the singular values are retained at each iteration. The entire embedding process took 24 seconds on average, with an average F1 macro score of 0.6573.

For all the methods used, the F1 scores are noticeably lower compared to the other examples, as indicated in Table 2.6. This may be a result of the structure of the network– the degree matrix for this graph is nearly singular. In particular, this causes the Matlab implementation of GraRep to produce embeddings that lead to inaccurate classifications, similarly to in the Amazon Co-Purchase example. However, in this case we still see that the commute time-based embeddings predict the department labels more accurately than most other methods. The SVD algorithm, in particular, had the highest F1 macro scores of all the methods, while also requiring less time than any other method except for the Matlab implementation of GraRep, which had a much lower F1 score.

Table 2.6 Email Dataset.

Method	d	**Runtime**	**F1 Macro**
Commute Times with SVD ($\alpha = 1$)	180	24 s	0.6573
DeepWalk	200	633 s	0.5169
GraRep (Python)	200	31.5 s	0.5431
GraRep (Matlab)	200	2.145 s	0.0048
node2vec (p=0.5, q=2.0)	200	2 min 20 s	0.6266
LINE	200	141 s	0.5423

2.5.5 COVID-19 lung image classifier

In this example, we apply the methods to a dataset of chest x-rays, saved as 299 $\times$ 299-pixel images. The dataset contains x-rays from patients with COVID-19, viral pneumonia patients, and healthy people, and is available from [8, 32]. In the following experiments we use 60 x-rays from each category. To apply the methods, we reshaped each image into a 299^2-dimensional vector, then treated these vectors as the initial representations of each x-ray. We construct the matrix T using the Euclidean norm between each two vectors to represent the similarity between those nodes. The commute time with SVD embeddings were based on 6 iterations of this algorithm, where 50% of the singular values are retained at each iteration.

For this dataset, we compared the F1 scores, averaged over 10 trials, obtained for this dataset using different amounts of training data. As shown in Table 2.7, the embeddings obtained from the method can be used to predict diagnoses on unlabeled data with some degree of accuracy, even with low amounts of training data. Using 90% of the dataset as training data, the method returns an F1 macro score of 0.8018 on average, while reducing the dimension of the vector representations from 299^2 to 3. As the amount of training data is decreased, the F1 macro score also decreases, but remains near 0.75 on average with only 50% training data. Even when training data is decreased to 10%, the average F1 score is above 0.5, at 0.5519.

Figure 2.5 shows the method's efficacy for clustering. In each subfigure, the low-dimensional embeddings show distinct grouping by color (here, yellow points represent images taken from patients with COVID-19, light blue represents images of healthy patients, and dark blue represents images from patients with viral pneumonia). The divisions between the clusters are particularly clear when a smaller dataset, containing only 20 points from each diagnostic category, is used.

Figure 2.6 shows the decrease in cross entropy loss over 10 iterations of SGD, for 10 different trials. In all trials, cross-entropy loss drops steeply for the first few steps of gradient descent, quickly converging to low values within 5 iterations.

Table 2.7 Lung X-Ray Dataset.

Method	d	**Runtime**	**F1 Macro**				
			Train $\% = 10\%$	30%	50%	70%	90%
CT w/ SVD	3	2.7 s	0.5519	0.6892	0.7441	0.7767	0.8018

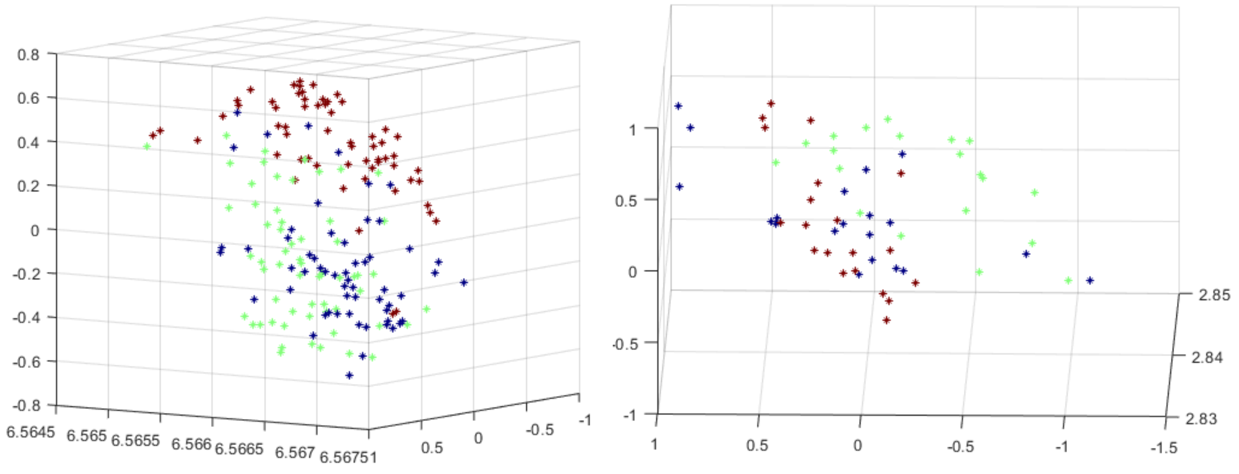

Figure 2.5 Three-dimensional node embeddings for the chest x-ray classifier, generated via the proposed method. Each color represents the diagnosis (normal, COVID-19, or viral pneumonia) associated with that x-ray.(a) shows the result of the method on a dataset consisting of 60 x-rays from each category, while (b) shows the result for a smaller dataset (20 from each category).

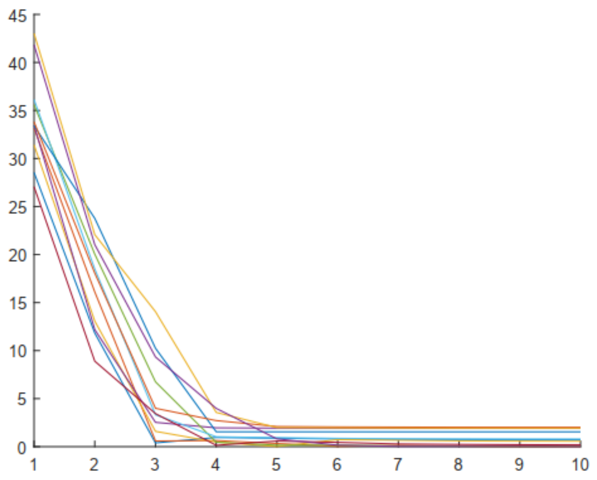

Figure 2.6 10 different trials of the embeddings for the chest x-ray classifier dataset. For each trial, the cross-entropy loss over 10 iterations of SGD is recorded. In all trials, the cross-entropy loss initially drops steeply, converging in 5 iterations.

2.6 Conclusion

This paper outlined a new method for embedding the vertices of a network while preserving commute time distances between nodes and optimizing cross-entropy loss. The method involves using truncated SVDs of a random walk matrix on the network to compute powers of the matrix, and via the Schultz method an approximate discrete Green's function, and then obtaining multiscale network embeddings from the SVD of a constant multiple of the product of the Green's function and the degree matrix of the random. As shown in Section 4, this method performs favorably on several datasets when compared with the existing methods DeepWalk [29], node2vec [18], LINE [39], and GraRep [6]. Future work could include a more complete analysis of the method, and possibly further improvements to the algorithm. In particular, a full analysis of the numerical linear algebra will be the subject of future investigation. Additionally, it is likely possible to improve upon the stochastic gradient descent used for decreasing cross-entropy loss here. The SGD stage also makes the largest contribution to computation time. Further improvements could involve incorporating other randomized algorithms to more efficiently optimize cross-entropy loss.

CHAPTER 3

ADAPTIVE NETWORK EMBEDDING WITH METADYNAMICS

3.1 Introduction

Many applications in biology and chemistry, including protein folding and chemical reactions, involve systems which behave according to some potential energy landscape. The potential energy of the system at any given time depends on its state at that moment–for example, the concentrations of reactants and products in a chemical reaction, or the arrangement of molecules in a particular protein's structure. In general, these systems tend toward the lowest energy states available to them, but due to the frequently complex nature of these energy landscapes, the behavior of these systems is not always that simple. Additionally, to fully understand the behavior of such systems, we also want to identify the mechanisms (e.g. transition states) by which a particular system moves from one minima to another.

A typical energy landscape for such a system may have many local minima, separated by energy barriers. In this case, we can view the system as a network of local minima, connected by edges weighted according to the energy barrier that must be overcome to transition between states. Taking this viewpoint allows us to utilize techniques for network embedding to analyze the behavior of these systems. One class of network analysis methods, node embedding methods, can be used to represent each node of a given network as a vector in a low-dimensional vector space, while retaining information about the relationships between nodes. Specifically, nodes that tend to co-occur in random walk simulations ("similar" nodes) will be embedded near each other in the vector space. Hence, these embeddings can be used to visualize the dynamics of complicated systems, and identify the transition paths by which systems change from one state to another.

However, another difficulty arises due to the presence of deep potential wells in these landscapes. While network embedding methods accurately identify energy minima where the system spends the majority of its time, other techniques are needed to get an improved, more detailed picture of the energy landscape. One such technique is metadynamics. Metadynamics uses a non-Markovian random walk to explore an energy landscape, where the energy landscape (and therefore the transition probabilities of the random walk) is adjusted by adding a Gaussian term after each step so that over time, the energy wells in the landscape will be filled up. As this happens, the process is discouraged from revisiting the lowest-energy states repeatedly, and the random walk spends more time exploring other areas of the landscape. The end goal is to have a flat landscape, from which we can recreate the original system by subtracting the additive terms.

In this article, we use network embedding techniques in combination with ideas from metadynamics to produce hierarchical embeddings that convey information about the system's behavior at different scales. Here we adjust the edge weights of the network in a way that parallels the original metadynamics over time to encourage exploration away from the lowest-energy minima. We show that these embeddings provide an effective way of visualizing the system's dynamics, including the entropy effect and internode relationships that may be obscured in other visualization methods.

The remainder of this article is structured as follows. In Section 2, we provide some background on network embedding, energy landscapes, and metadynamics. In Section 3, we discuss Lennard-Jones clusters, and demonstrate our methodology on these test systems. Section 4 contains an application to a more complicated system: DNA folding in a human telomere.

3.2 Background

3.2.1 Network Embedding

Real-world networks, particularly those representing biological and chemical systems, are often large and have complex structures, making them difficult to work with. To improve understanding of these networks, the nodes of a given network can be embedded into a low-dimensional vector space. There exists a wide variety of methods for network embedding, which may use random walks, matrix factorization, deep learning, or some combination of the three. The idea behind all these methods is that the nodes in a given network can be mapped into a low-dimensional vector space in such a way that closely related nodes have similar embeddings in the embedding space.

One broad category for node embedding techniques involves the use of a random walk on the network. This typically means building a random walk such that the possible states of the walk are the nodes of the network, and the transition rates are determined by the edges and edge weights.

In methods such as DeepWalk and node2vec [29, 18], the general approach is to use short random walk simulations to determine similarities for each pair of nodes, then encode this information into embeddings via gradient descent. Two nodes are similar if there is a high probability that a random walk simulation containing one node will also contain the other. Then, the embeddings are assigned based on these conditional probabilities using the SkipGram algorithm, optimizing the loss function

$$Loss = \sum_{i} \sum_{j \in walk(i)} - \log\left(P(j|i)\right).$$

where $walk(i)$ denotes a random walk trial starting at node i, and $P(j|i)$ denotes the conditional probability that a random walk starting at node i will include node j, modeled by (where v_i denotes the embedding of i)

$$P(j|i) = \frac{\exp(v_i \cdot v_j)}{\sum_k \exp(v_i \cdot v_k)}.$$

The efficiency of this optimization can be greatly improved by using negative sampling [26], which instead models the probabilities as:

$$P(j|i) = \sigma(v_i \cdot v_j) \prod_{m=1}^{M} \sigma(-v_i \cdot v_{k_m}),$$

where $\sigma(x) = 1/(1 + e^{-x})$ and M is the chosen number of negative samples (sampled from a uniform or unigram distribution).

Some other random walk-based methods utilize matrix factorization of a random walk matrix, rather than performing random walk trials. In this article we use one such method, which we elaborate on in the next section.

3.2.1.1 Embedding Method

In the examples here, the embeddings are produced via a sparse approximation-based embedding method introduced in Chapter 2. The algorithm is as follows:

First, we produce compressed approximations to the dyadic powers of the random walk on the network using its principal components; taking U_k and Σ_k to be the top left singular vectors and singular values of $T_{k-1} \approx T^{2^{k-1}}$, we let $T_k := U_k^T T_{k-1} U_k$.

From this, we produce a low-rank approximation to the Green function of the random walk, using the Schultz method:

$$\begin{aligned} Gf &= \sum_{k=1}^{\infty} T^k f = \prod_{k=0}^{\infty} (I + T^{2^k}) f \\ \Rightarrow \hat{G} &= \prod_{k=0}^{K} (I + T_k). \end{aligned}$$

The embedding matrix θ that satisfies $\theta^T \theta = vGD^{-1}$, where v gives the volume of the graph and D its degree matrix, is a commute time embedding [19]. Denoting the leading singular values and left singular vectors of the matrix $v\hat{G}D^{-1}$ by Σ_G and U_G, we take $\theta^T := U_G \Sigma_G^{1/2}$.

Using this as a starting point, we introduce parameters by multiplying each column j of θ by a weight C_j, and optimize these weights to minimize crossentropy loss via SGD.

3.2.2 Energy Landscapes

For many biological and chemical applications, the potential energy landscape provides insight into the dynamics of the system. Statistical analysis of the potential energy surface has been an active field of research for this reason, leading to results regarding global optimization, characterization of local minima and transition states, and other ways of gathering information about the energy landscape. These landscapes can be used to both understand the behavior of a system, and to improve simulation techniques. Throughout the existing methods for energy landscape analysis, the central idea is to identify local minima via geometric optimization, and to find transition states connecting them via steepest-descent pathways [40].

One method for energy landscape exploration is basinhopping, introduced in [42]. Basinhopping is a Monte Carlo method that produces a database of local minima within an energy landscape. The basic idea of basinhopping is to simplify the energy landscape by mapping each point in the energy landscape to the local minimum it is closest to, and then constructing a random walk between these basins. In other words, a basinhopping run starts at a point on the energy landscape and performs a geometric optimization until a local minimum is reached. Then, each coordinate is shifted by a random amount, and another geometric optimization is done on the new point.

3.2.3 Metadynamics

Metadynamics was introduced in [27] as a technique to aid in the exploration of energy landscapes. The central idea is to create a non-Markovian (and in a sense, self-avoiding) random walk by adjusting the gradient of the energy landscape after each step by addition of a Gaussian term. Over time, these Gaussians eventually fill up the valleys in the enegy potential, which allows the random walk to explore other areas of the landscape and leads to a more complete picture of the system dynamics. Specifically, after each step, the parameter ϕ_i, which represents the derivative of the energy with respect to the ith parameter $-\frac{\partial E}{\partial \sigma_i}$, is adjusted according to:

$$\phi_i^{t+1} = \phi_i^t - \frac{\partial}{\partial \sigma_i} W \prod_i \exp(-\frac{|\sigma_i - \sigma_i^t|^2}{2\delta\sigma^2}),$$

where W and $\delta\sigma$ are the height and width of the Gaussian respectively, to be chosen based on prior knowledge of the energy landscape.

The main goal of this note is to apply the central idea of metadynamics to network embedding. In particular, we use an iterative embedding process, adjusting the energy landscape at each iteration, to get a series of hierarchical embeddings that provide more information about the full energy landscape than a single embedding by itself. Since the energy landscape is represented by a network via an adjacency matrix A, at each iteration we update the entries of A according to:

$$A(i,j) = A(i,j) - W \prod_i \exp(-\frac{||\theta_i - \theta_j||^2}{2\delta\sigma^2}). \tag{3.1}$$

3.3 Lennard-Jones clusters with metadynamics

Lennard-Jones clusters are a test system used to model atomic or molecular dynamics within a fluid. While they are far simpler than more detailed models that portray these dynamics more realistically, their simplicity makes them an ideal starting point for global optimization and energy landscape analysis. Optimization techniques used with Lennard-Jones clusters may eventually provide insight into more complex optimization problems involving potential energy fields, such as protein folding, and the dyamics of more accurate fluid models.

In Lennard-Jones clusters, we assume that the potential energy E of a given configuration of atoms depends only on distances between atoms:

$$E(r) = 4\epsilon \sum \left[\left(\frac{\sigma}{r_{ij}} \right)^{12} - \left(\frac{\sigma}{r_{ij}} \right)^{6} \right]$$

where r_{ij} denotes the distance between atoms i and j, and the parameters σ and ϵ represent pair equilibrium separation and well depth. In the experiments below, we utilize reduced units (e.g. $\sigma = \epsilon = 1$).

3.3.1 Adjacency matrix-based embeddings

To analyze the following Lennard-Jones clusters, we first generated a database of local energy minima using the Pele software available at [41]. The database was produced using a basinhopping run of 500 steps, and consists of 8797 local minima connected by 8099 transition states. The database also contains other thermodynamic information, including the potential energy at each local minimum. The embeddings here are based on a network we construct with nodes given by the local minima, and edges located between each pair of nodes where a transition state has been identified, weighted according to the potential energy.

In the following examples, we used the adjacency matrix for this network to construct the embeddings, with entries given by the energy barriers between states. Initially, every node is embedded into the vector space. Since points on the transition path should be near the global minimum in terms of commute time, this typically results in most of the nodes being clustered around the global minimum's embedding. Then, re-embedding only the nodes in that cluster reveals new, more detailed information about some of the remaining nodes. We do this by creating a new, smaller adjacency matrix including only those points, and adjusting the edge weights according to the Gaussian term from Section 2.3. Iterating this process leads to a sequence of embeddings, each of which provides a visualization of the energy landscape at a different scale. These embeddings can also be used to investigate the effects of entropy on a given system, here demonstrated by comparing embeddings for the same system at different temperatures.

3.3.1.1 8 Atom Lennard-Jones Embeddings

Figures 3.1 (a) and (b) represent the disconnectivity tree and embeddings (before and after applying the metadynamics adjustment in Equation 3.1) for the 8-atom Lennard-Jones cluster. The colors on each minimum on the disconnectivity tree are matched to those nodes' embeddings in Figure 3.1, with some of the nodes, e.g. those colored red, embedded to the same place. In the embeddings, nodes with closer dynamic relationships are clustered together, since they will have the smallest commute time distances between them. However, visual inspection of the embeddings reveals that they also seem to be clustered according to potential energy levels. Nodes that have similar energy levels (such as the three highest-energy nodes, for example) have been embedded almost indistinguishably near to each other. On a larger scale, we can also see how the higher energy nodes relate to the nodes with the two lowest energies. Specifically, the global minimum (in dark blue) is nearer to the nodes in the middle of the tree (in orange), while the second-lowest local minimum is much nearer to the three highest energy nodes. This allows us to draw conclusions about a lowest-commute-time path through these states: one of the orange-colored states might transition directly to the global minimum, while one of the higher-energy states would be more likely to transition to the second-lowest-energy state first, and then either remain there or transition on to the global minimum.

Just as the temperature of a molecular system affects that system's behavior, embeddings obtained by this method will change with the introduction of entropic forces. Figure 3.2 shows the change resulting from an upward shift in temperature for this 8-atom cluster. The major change seen between these two temperatures is that at the higher temperature, the nodes' embeddings are more spread out–closely related nodes are embedded near each other, but not indistinguishably so, as in Fig. 3.2 (a). Although the overall groupings are similar in most respects, there are some changes. in particular, one of the higher-energy nodes (in red) is much more distanced from the others at high

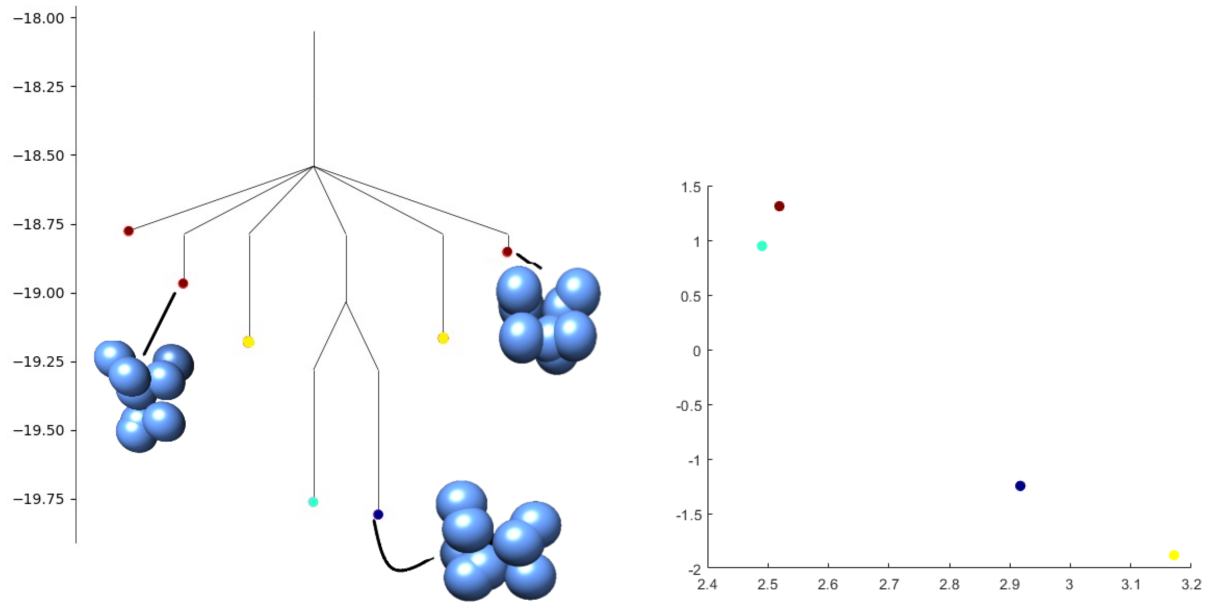

Figure 3.1 Disconnectivity tree and metadynamics-based embeddings for the Lennard-Jones cluster with 8 atoms. Left: Disconnectivity tree of all local minima. Right: Embeddings for the local minima after applying the adjustment in 3.1. Closely related minima here have very similar or identical embeddings, for example both yellow minima are embedded at the yellow point on the right. In all figures, color scheme denotes the potential energy of the system at the configuration represented by that node. Dark blue denotes low energies (e.g. the global minimum), and red denotes the highest-energy states.

temperatures. At lower temperatures, the higher-energy nodes were all approximately equidistant from the two lowest-energy configurations, but at higher temperatures, we see that most of the nodes are much closer to one of the lowest-energy states, and further from the other.

For comparison, we also investigated the inter-node relationships by directly computing the commute times between each pair of nodes. The commute time between nodes i and j, or mean time for the Markov process on the graph prescribed by the Laplacian L to travel from i to j and back, is given by

$$CT(i,j) = vol \sum_{k=2}^{N} \lambda_k(\phi_k(i) - \phi_k(j))^2,$$

where $0 = \lambda_1 \leq \lambda_2 \leq \lambda_3 \leq ... \leq \lambda_N$ are the eigenvalues of L, and the ϕ_k are the corresponding

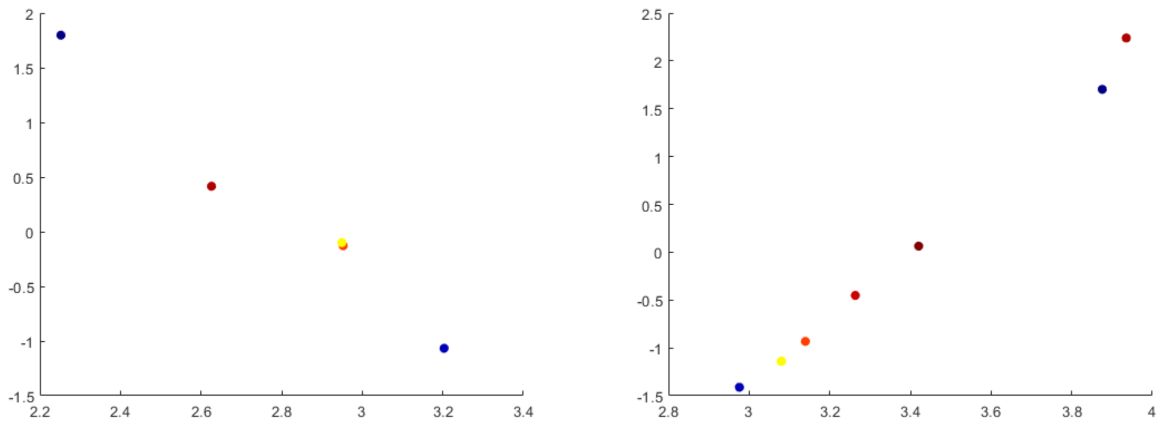

Figure 3.2 Metadynamics-based embeddings for the 8-atom cluster at the temperatures $T = 0.08$ (left) and $T = 1$ (right). Color scheme denotes the potential energy of the system at the configuration represented by that node. Dark blue denotes low energies (e.g. the global minimum), and red denotes the highest-energy states.

eigenvectors. The off-diagonal entries for the Laplacian for a Lennard-Jones cluster with κ degrees of freedom are given by

$$L(i,j) = \sum_k \frac{O(i)}{O_k(k)} \frac{v(i)}{v_k(k)}^{\kappa-1} v(i) \exp(-\beta(E_k(k) - E(i))),$$

where O and O_k represent point group orders of the local minima and transition states, respectively, and similarly v and v_k represent mean vibrational frequencies and E and E_k represent potential energy levels at each configuration.

3.3.1.2 Lennard-Jones 38 Cluster Embeddings

Figure 3.5 shows the hierarchical embeddings for the 38-atom Lennard-Jones cluster. The disconnectivity tree [40] for this system is a little more complex, but we can still apply the same reasoning to compare it to the embeddings.

Here, we used the adjacency matrix for this network to construct the embeddings, with entries given by the energy barriers between states. The first image shows the initial embeddings, colored

Table 3.1 Commute times between nodes, 8-atom Lennard-Jones cluster.

Node 1	**Node 2**							
	1	2	3	4	5	6	7	8
1	0	25.4	17.7	23.5	29.0	65.2	86.1	88.4
2		0	7.7	13.5	19.0	55.2	60.7	63.0
3			0	5.8	11.3	47.5	68.4	70.7
4				0	17.1	53.3	74.3	76.5
5					0	58.8	79.7	82.0
6						0	115.9	118.2
7							0	123.7

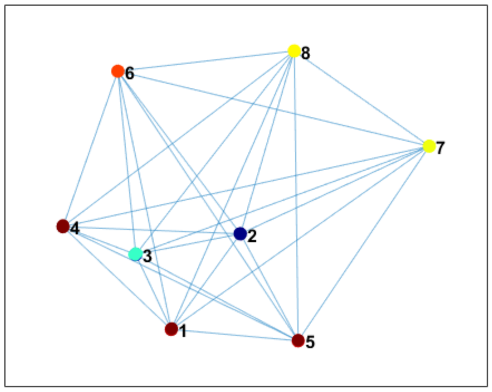

Figure 3.3 The 8-atom Lennard-Jones network of local minima. Edge lengths are proportional to commute times. Node colors are chosen to match those in Figures 3.1 (a) and (b).

by their commute time distance from the global minimum. We re-embedded the points of interest (the nodes nearest to the origin) and used two iterations of this process to obtain the second image.

In the first stage embeddings (Figure 3.5 (a)), the nodes are arranged according to commute time distance from the global minimum, which corresponds loosely to potential energy level. In particular, the nodes with $E > -170$ (which correspond to the highest-energy clusters in the tree) are embedded further from the global minimum. In the next embedding, these nodes are removed due to their distance from the global minimum, and the more central cluster will be re-embedded. As we consider the second-stage and later embeddings, this pattern continues: each re-embedding reveals a new "layer" of nodes that are embedded further from the global minimum, and which correspond to nodes on the disconnectivity tree that are lower-energy than the nodes removed in the previous stage, but higher-energy than the more closely clustered nodes. In particular, at the second level (Fig. 3.5 (b)), the embeddings have 3 small "spokes" originating from a central cluster containing the global minimum. However, these embeddings provide additional context: nodes that are embedded within a particular "spoke" are more closely related, which means the system is more likely to transition between these states. Since the nodes do not all come from the same group on the disconnectivity tree, these embeddings can also reveal interactions between nodes that aren't indicated on the disconnectivity tree. Additionally, since the spokes are connected to the cluster containing the global minimum, we can conclude that each spoke represents a potential transition pathway from the outer edge of the cluster to the center. In other words, if the system is at a state represented by the outer point of one of the spokes, its most likely path toward the global minimum will be to travel through the states represented by other nodes in the spoke. We can apply similar reasoning to higher-level embeddings. Each stage reveals a more detailed picture of the dynamics of a different part of the system's energy landscape. The first levels give a coarse-grained picture, only identifying broad groups of high-energy and low-energy nodes, while later levels give a more fine-grained visualization of the nodes most closely related to the global minimum.

Often, we want a more detailed, finer-grained visualization of the energy landscape than the disconnectivity tree in Figure 3.4 can provide. It is informative, therefore, to re-embed the parts of the network of greatest interest to gain further insight. Here we focus on the lowest-energy parts of the Lennard-Jones energy landscape. We repeated the above process using the subnetwork consisting only of the nodes with potential energy < -170.9, that is, the 163 lowest-energy nodes. Figure 3.6 shows the results of this experiment.

Nodes in these embeddings (both Figure 3.5 and Figure 3.6) are clustered according to their similarity in terms of commute time. In other words, nodes that can be quickly and frequently reached from either the global minimum or second lowest energy node will be grouped near them. As a result, most of the nodes we are most interested in end up in the same cluster, and the embeddings

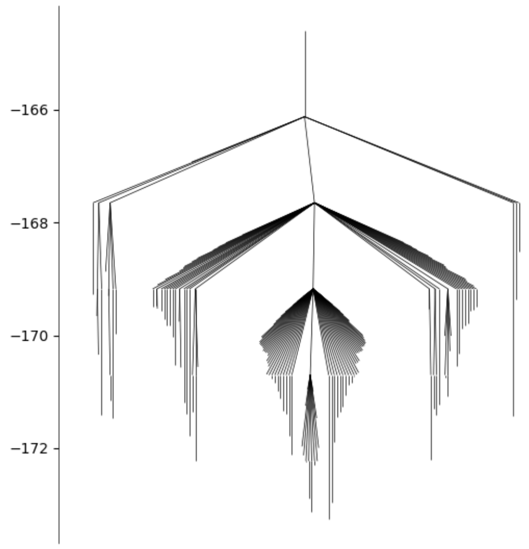

Figure 3.4 Disconnectivity tree for the 38-atom Lennard-Jones cluster.

obtained from the first application of the embedding method only give useful clusters for nodes that are more distant in terms of commute time from the part of the graph of interest. As we progress through additional iterations, we pull apart the cluster containing the global minimum, positioned near the origin in the first iteration's embeddings. The final iteration's embeddings give us a clear picture of the dynamics of the process within the subnetwork defined by the nodes within this cluster.

If two nodes share a cluster or are close in the embedding space, this indicates that the system can easily transition between those nodes, with a relatively low energy barrier. As a result, the groupings seen in these embeddings correspond to the groupings in the disconnectivity tree for this system [40, 41]. For instance, one of the clusters in the second-stage embeddings correspond to the global minimum and its nearest neighbors (pictured directly right of center in the disconnectivity tree in Figure 3.4). Clusters can be mapped to the disconnectivity tree by comparing the potential energies of the nodes within the cluster to the tree.

In the third iteration, some of the clusters instead represent combinations of multiple disconnec-

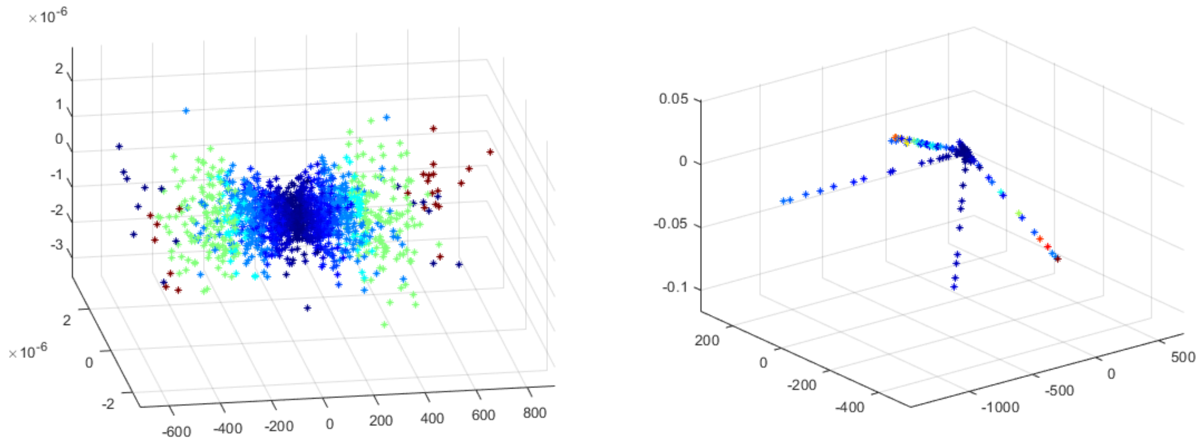

Figure 3.5 Hierarchical embeddings for the Lennard-Jones cluster with 38 atoms. Pictured are the embeddings for the local minima originally clustered at the origin (with the global minimum) after applying metadynamics (equation 3.1). Color scheme denotes commute time from the global minimum, with dark blue denoting short distances, and red denoting the furthest distances.

tivity tree groups; this is a result of re-embedding the nodes to spread out those nodes that were previously near the origin–those nodes end up being embedded in or near the clusters they are most closely related to, although they are not actually members of the respective tree groupings. For instance, the global minimum is embedded directly next to a node from a neighboring tree group.

So, the first iteration's embeddings tell us about higher-energy nodes and those that are more distant from the global minima, and further iterations reveal information about parts of the graph nearer and nearer to those minima.

Additionally, these embeddings may be useful for identifying transition paths. The structure of the second-stage embeddings, in particular, reveals four transition paths– if the system is initialized from a node near the outside of one of these "spokes", its lowest-energy path to the global minima will involve following the spoke into the center cluster.

Adding up the distinct clusters in Figure 3.6, we see that we have identified about as many clusters as there are clusters on the disconnectivity tree. However, the clusters in the embeddings do not map directly onto tree clusters– clusters in the embedding space frequently contain closely-related nodes from neighboring groups on the tree. To get a better idea of how these embeddings compare to the disconnectivity tree, we can consider the potential energy levels of each local minima.

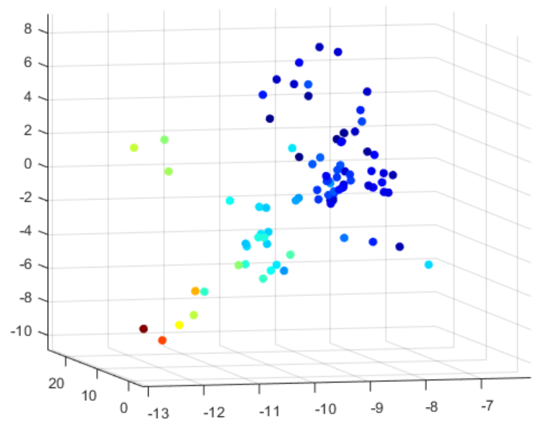

Figure 3.6 Embeddings of the local minima of the 38-atom Lennard-Jones cluster with potential energies less than -170.9. The left figure shows the original embeddings of all 163 minima; the right figure shows stage 2 embeddings (embeddings of the nodes clustered around the origin, after adjusting the adjacency matrix according to the metadynamics adjustment in equation (3.1)). Color scheme denotes commute time from the global minimum, with dark blue denoting short distances, and red denoting the furthest distances.

We can also use these embeddings to observe the results of entropic changes to the system. In Figure 3.7, the embeddings for this cluster under two additional temperature conditions are shown. The results of the temperature change are comparable to what was observed previously with the 8-atom cluster. Namely, at higher temperatures (Fig. 3.7 (b)) , there is greater variation in the node embeddings, while at lower temperatures (Fig. 3.7 (a)) closely related nodes are more likely to embedded much nearer to each other, creating the impression that there are fewer embeddings pictured.

3.4 Application: Human Telomere Folding

Here we attempt to apply this method to a more complex problem: DNA folding in a human telomere. Specifically, we consider a sequence of 22 nucleotide bases $A(G_3TTA)_3G_3$ which repeats within human telomeres. This sequence is known to form a G-quadruplex, a type of secondary structure formed by groups of four guanine bases called G-tetrads. Its structure and potential energy

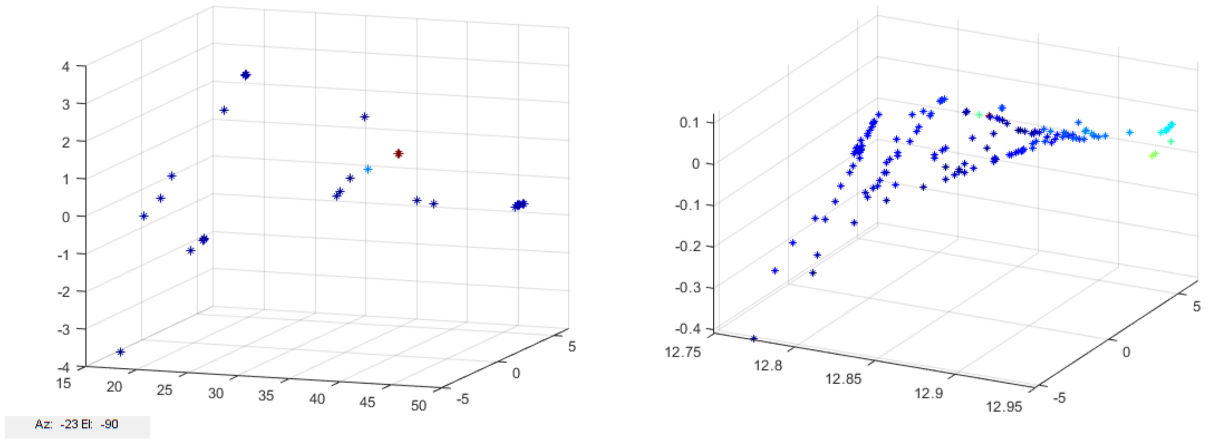

Figure 3.7 Metadynamics-based embeddings for the 38-atom cluster at the temperatures $T = 0.08$ (left) and $T = 1$ (right). Color scheme denotes the commute time distance from the global minimum. Dark blue denotes low distances (e.g. the global minimum), and red denotes the highest.

landscape were previously investigated in [12].

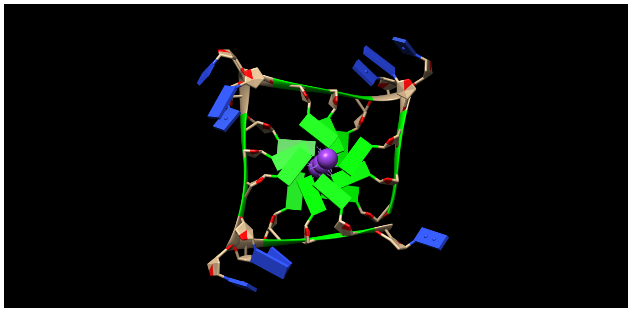

Figure 3.8 The four-strand G-quadruplex structure (PDB structure 1KF1), with guanine nucleotides colored green. Image produced with Chimera [31].

In [12], the potential energy landscape was modeled using the HiRE-RNA model for coarse-grained DNA modeling [28], which involves a summation of energy terms representing three different types of bonds between atoms: local interactions, non-bonded interactions, and hydrogen bonds. For each of the 22 nucleotides in the telomere, six or seven atoms are considered by the model. Discrete path sampling was then used to generate a database of local minima and their connecting transition states.

We use this database as a starting point. In particular, we construct a network with nodes given by the 4000 lowest-energy local minima, and edges between nodes determined by the transition states connecting them.

For these first experiments, we used a random walk based on the energy barriers between states, as in Section 3.1. Figure 3.9 shows the results of the initial embeddings and the second, third, and fourth iterations, based on an energy landscape adjusted by a Gaussian term with width 0.75 and height 1 after each successive embedding. As in the Lennard-Jones cluster example, the first embedding displays embeddings of all nodes, while each successive embedding shows a re-embedding of the subnetwork of nodes most closely related to the global minimum (that is, all nodes whose previous embeddings lie within a small tolerance of the global minimum).

As with the Lennard-Jones clusters, each level of embedding represents "zooming in" on the part of the network around the global minimum. The first level gives us a global view – the global minimum and its nearest neighbors (with respect to commute times) are clustered at the origin, represented by a dark blue dot. The red and orange dots furthest away from the origin represent local minima which are more distantly related, requiring multiple steps or higher energies to transition to the global minimum. Potential transition paths can be identified by starting at one of these points, and moving toward the origin along nearby points. At the second level (and each of the following levels), the nodes embedded closest to the local minimum are re-embedded to give us a more detailed insight into the relationships of those nodes. The potential transition paths here can be constructed similarly.

3.4.1 Combining Metadynamics with Transition Path Theory

For energy landscapes with a large number of local minima, the corresponding networks contain large numbers of nodes. In fact, for the Lennard-Jones clusters, the number of minima increases exponentially with the number of atoms. In these situations, as we have seen, it is beneficial to have a method for network embedding that focuses on locally embedding regions of the graph that are of particular interest, for example the subnetwork consisting of the global minimum and its nearest

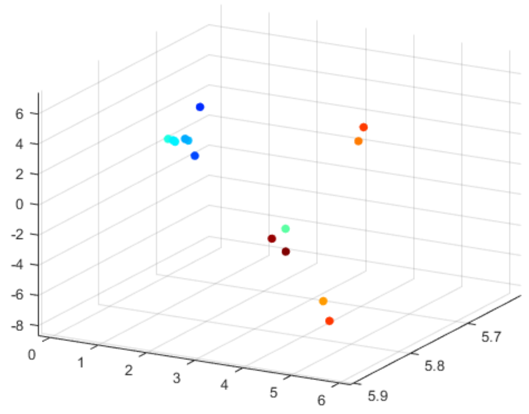

Figure 3.9 Embeddings of the local minima network for the human telomere sequence, based on the adjacency matrix as in section 3.1 and the metadynamic adjustment in 3.1. Colors of embeddings denote distance from the global minimum, with dark blue denoting the shortest distances and red the furthest.

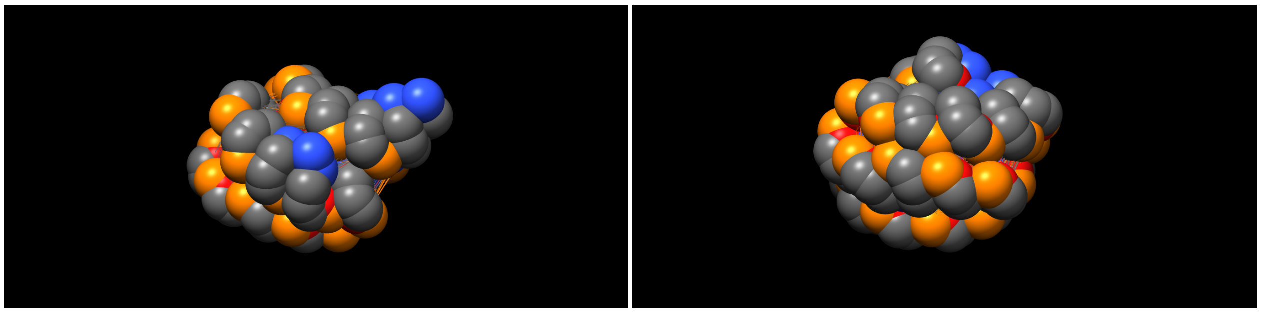

Figure 3.10 Two local minima from the telomere's energy landscape, with potential energies of -68.0 and -69.7 respectively. Images produced with Chimera [31].

neighbors. Since the states in this subnetwork are embedded near each other, the commute times between them are low, so the time required for the system to move between these configurations is low. The states in these subnetworks often represent molecular configurations that are similar, differing only by a simple change; therefore these subnetworks can have dramatically fewer degrees of freedom compared to the full network.

We next present an alternate formulation, which replaces the adjacency matrix with the probability current matrix from transition path theory (TPT). TPT is a theoretical framework that can be used to study transition events (such as a molecule's transition from one configuration to another). In particular, TPT characterizes the transition paths of a system based on statistical properties such as rate functions and probability currents which measure the average rates of transitions between states. For large networks, particularly those with irregular structures, the committor functions needed to apply transition path theory to the full network may prove difficult or impractical to compute. Focusing our application of TPT onto a smaller, localized subnetwork avoids this difficulty, allowing us to take advantage of the additional information TPT offers. In the following experiments, we first embed all the local minima using the adjacency matrix, as in the previous section, then construct a subnetwork representing the telomere energy landscape nearest to the global minimum using a local transition path theory. Taking $S' \subset S$ to be the set of all nodes which are embedded within distance δ from the global minimum, we construct a subnetwork consisting only of the nodes in S' by considering only the relevant portions of the adjacency matrix (adjusted according to the Gaussian term 3.1). Then, we compute the graph Laplacian $L = D - A$, and compute the committor functions and probability currents as follows.

The forward and backward committors q^+ and q^- solve the system

$$\begin{cases} \sum_{j \in S} L_{ij} q_j^+ & = 0 \qquad \forall i \in (A \cup B)^C \\ q_i^+ & = 0 \qquad \forall i \in A \\ q_i^+ & = 1 \qquad \forall i \in B. \end{cases}$$

In these experiments, I took the global minimum to be the initial state A, and chose the local minimum with the highest associated potential energy to be B.

To find the probability current f and effective probability current f^+,

$$f_{ij} = \begin{cases} \pi_i q_i^- L_{ij} q_j^+, & \text{if } i \neq j \\ 0 & \text{if } i = j, \end{cases}$$

and

$$f_{ij}^+ = \max(f_{ij} - f_{ji}, 0).$$

Finally, each subnetwork is embedded into $\mathbb{R}^3$ using the method described in Section 2.1.1, using the effective probability current in place of the adjacency matrix. Here we apply the hierarchical embedding procedure to the subnetwork of nodes surrounding the global minimum (as we have done throughout this paper), but the same process could be used to examine other areas of the graph aside from the global minimum to obtain a fuller overall picture of the energy landscape.

The distances between nodes within each subnetwork preserve the commute times between those nodes. Since the cross-entropy loss minimization is applied to the overall network at each step, we can expect that the distances between nodes correspond to likelihoods of a transition between those nodes, similarly to the previous examples, and therefore we can interpret the embeddings and identify possible transition paths in the same way.

Though the main purpose of applying TPT in this way is to analyze larger networks, we can briefly demonstrate its efficacy on a smaller system–the 8-atom Lennard-Jones cluster from Section 3 (see Figure 3.11). We still see that the two nodes with potential energies near -19.2 (colored

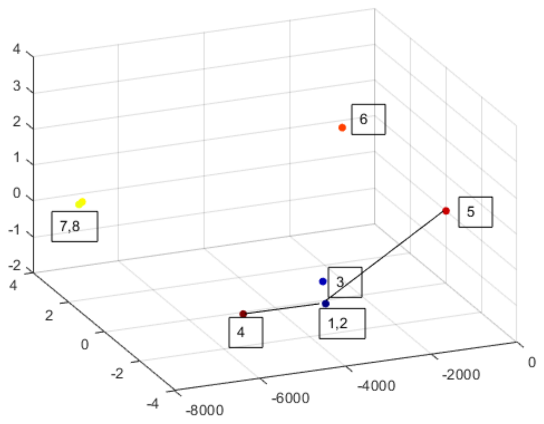

Figure 3.11 Hierarchical embeddings of the local minima network for the 8 atom LJ cluster, based on a TPT-based subnetwork consisting of the nodes clustered around the global minimum (in dark blue), and the metadynamic adjustment in 3.1. Colors of embeddings denote potential energy at each configuration, with dark blue denoting the lowest energies and red the highest.

yellow in Figs 3.1 and 3.3) are closely connected, and the nodes represented in red and orange are closer to the lowest energy nodes than to each other, but in this case the short distance between the

2 lowest-energy nodes reflects a higher transition rate between them. The highest-energy nodes, embedded in red, are also embedded separately in this case, whereas their adjacency matrix-based embeddings were identical. These embeddings indicate that a transition path (shown in Fig. 3.11) from the highest-energy node to the slightly lower-energy node labeled 5 might pass through nodes 1 and 2 on the way. Direct simulations of this system confirm this transition path. Hence, these embeddings can be useful for predicting the mechanisms by which a molecule moves between two configurations.

Having demonstrated our method on a simpler system, we now return to the human telomere molecule. Figure 3.12 shows the embeddings produced via the TPT process described above. The colors of local minima in Figure 3.12 are determined by the commute times between those nodes and the global minimum, which is embedded in dark blue. It is immediately apparent that the local minima are grouped according to these commute times, with separate clusters containing most of the red, yellow, and light blue nodes. Within each of these clusters, the nodes are relatively close in terms of commute times, indicating that these molecular configurations are similar up to some simple molecular change. As with the 8-atom Lennard-Jones cluster, these embeddings suggest possible transition paths. For example, if the system starts at one of the states furthest from the global minimum (colored in red, clustered in the upper left) one would expect a transition path to the global minimum to travel through the light blue and yellow clusters to reach the node in dark blue.

3.5 Conclusions

This article presents a novel method for adaptive, hierarchical network embedding that combines the ideas of metadynamics with node embedding techniques. The usefulness of metadynamics for analyzing energy landscapes can then be extended to energy landscapes modeled by networks and other network applications. In this embedding scheme, the network itself – both its edge weights and the set of nodes under consideration – can be adjusted to more effectively focus on particular areas of the graph. In the context of energy landscapes, this allows us to focus on important parts of the graph, such as the lowest-energy minima and the nodes most closely connected to them. The same idea could be adapted to focus on other features of a given network as well. In some cases,

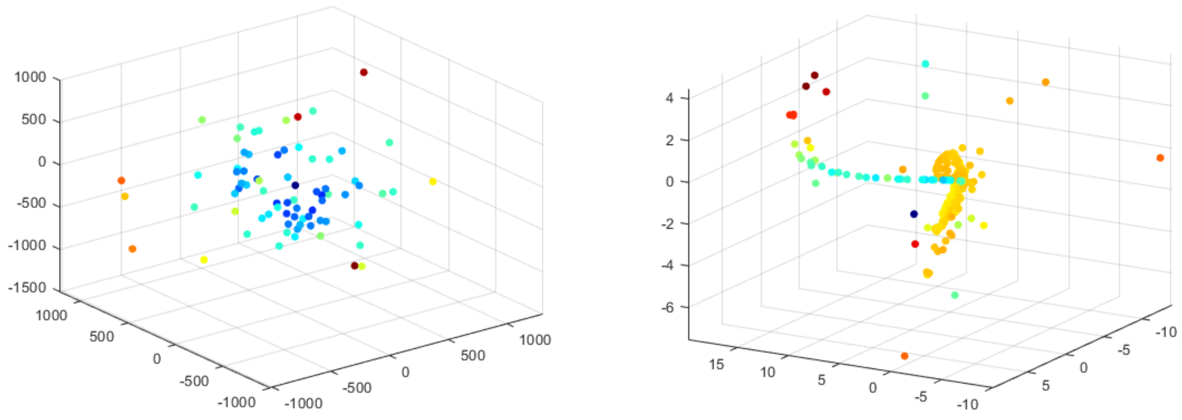

Figure 3.12 Hierarchical embeddings of the local minima network for the human telomere sequence, based on a TPT-based subnetwork consisting of the nodes clustered around the global minimum (in dark blue), and the metadynamic adjustment in 3.1. Colors of embeddings denote commute time distance from the global minimum, with dark blue denoting the shortest distances and red the furthest.

the embeddings illuminate aspects of the system's dynamics like temperature dependence that are not explored in other methods, such as disconnectivity graphs. To demonstrate the success of this method, we present the resulting hierarchical embeddings associated with a Lennard-Jones cluster of 38 atoms, as well as a more complicated system representing a DNA sequence from a human telomere. In the future, further work in this direction may lead to a scheme for analyzing larger, more complex biochemical systems such as protein folding. We also plan to develop a localized form of TPT, which could be applied to locally parts of a graph in order to make TPT more accessible for larger-dimension problems.

APPENDIX

APPENDIX

PROOFS OF THEOREMS

Theorem 1

Theorem. Let S be the set of all vertices in a network with unnormalized Green function given by G, degree matrix D, and volume vol, and let $CT(\cdot, \cdot)$ denote the commute time. There exists $\Pi \in \mathbb{C}^{m \times d}$ where $m = O(\log(1/\delta)/\varepsilon^2)$ and $\varepsilon > 0$ such that for all $x, y \in S$,

$$(1-\varepsilon)CT(x,y) \leq ||\Pi(volGD^{-1})^{1/2}\delta_x - \Pi(volGD^{-1})^{1/2}\delta_y||_2^2 \leq (1+\varepsilon)CT(x,y) \qquad \text{(A.1)}$$

with probability $1-\delta$.

Proof. If we let $\lambda_0, \lambda_1, ..., \lambda_{|S|}$ and $\zeta_0, \zeta_1, ..., \zeta_{|S|}$ denote the eigenvalues and eigenvectors of a matrix T, the diffusion distance at time t induced by T is defined by

$$d^{(t)}(x,y) = \sqrt{\sum_\lambda \lambda^t(\zeta_\lambda(x) - \zeta_\lambda(y))^2} = ||T^{t/2}\delta_x - T^{t/2}\delta_y||_2.$$

Note that G is a left inverse of the random walk Laplacian L (that is, $GL = I$ so long as the graph is connected), and so the eigenvalues of G are the reciprocals of the eigenvalues of L. (G and L also have the same eigenvectors.) Let $\lambda_0, \lambda_1, ..., \lambda_{|S|}$ and $\zeta_0, \zeta_1, ..., \zeta_{|S|}$ denote the eigenvalues and eigenvectors of the normalized Laplacian $\mathcal{L}$. Then

$$CT(x,y) = vol\sum_{i=2}^{|S|}\frac{1}{\hat{\lambda}_i}(\frac{\zeta_i(x)}{\sqrt{D(x,x)}} - \frac{\zeta_i(y)}{\sqrt{D(x,x)}})^2 \qquad \text{(A.2)}$$

$$= \sum_{i=2}^{|S|} vol\lambda_i(\frac{\zeta_i(x)}{\sqrt{D(x,x)}} - \frac{\zeta_i(y)}{\sqrt{D(x,x)}})^2 \qquad \text{(A.3)}$$

$$= ||(vol \cdot GD^{-1})^{1/2}\delta_x - (vol \cdot GD^{-1})^{1/2}\delta_y||_2^2, \qquad \text{(A.4)}$$

where the final equality comes from the fact that the eigenvalues of $vol \cdot G$ are given by $vol\lambda_i$ for all i, and by noticing that $\sum_{i=2}^{|S|} vol\hat{\lambda}_i(\zeta_i(x) - \zeta_i(y))^2$ is equal to the square of the diffusion distance for $t = 1$, using the eigendecomposition of $vol \cdot G$.

Then, let $\Pi = \frac{1}{\sqrt{m}} H$ where H has random entries $h_{ij} \sim N(0,1)$. Fix $x, y \in V$. We can apply the original Johnson-Lindenstrauss lemma to conclude that

$$(1-\varepsilon)||(vol \cdot GD^{-1})^{1/2}\delta_x - (vol \cdot GD^{-1})^{1/2}\delta_y||_2^2 \leq ||\Pi(volGD^{-1})^{1/2}\delta_x - \Pi(volGD^{-1})^{1/2}\delta_y||_2^2 \tag{A.5}$$

$$\leq (1+\varepsilon)||(vol \cdot GD^{-1})^{1/2}\delta_x - (vol \cdot GD^{-1})^{1/2}\delta_y||_2^2, \tag{A.6}$$

and therefore

$$(1-\varepsilon)CT(x,y) \leq ||\Pi(volGD^{-1})^{1/2}\delta_x - \Pi(volGD^{-1})^{1/2}\delta_y||_2^2 \leq (1+\varepsilon)CT(x,y) \tag{A.7}$$

with probability $1-\delta$. □

Theorem 2

Theorem. Let S be the set of all vertices in the network, $|S| = N$, and let $CT(\cdot,\cdot)$ denote the commute time.

1. If $\Phi(x) := \Phi^T\delta_x$ denotes the embedding for node x obtained by applying the diffusion wavelet algorithm to the matrix $volGD^{-1}$ with precision constant ε (that is, Φ is the orthonormal basis of scaling functions and $\Phi(x)$ is the xth row of Φ), then for all $x, y \in S$,

$$-\varepsilon + \sqrt{CT(x,y)} \leq ||\Phi^T(\delta_x - \delta_y)|| \leq \varepsilon + \sqrt{CT(x,y)}. \tag{A.8}$$

2. Let $U\Sigma V^T \approx volGD^{-1}$ represent the truncated SVD using the αN leading singular values $\sigma_1 \geq ... \geq \sigma_{\alpha N}$, for some $\alpha \leq 1$. Let Θ denote the embedding matrix obtained from the αN leading left singular vectors U, such that $\Theta = (\Sigma^{1/2})^T U^T$. Then for all $x, y \in S$, and $\varepsilon = O(\sigma_{\alpha N+1})$,

$$-\varepsilon + CT(x,y) \leq ||\Theta^T(\delta_x - \delta_y)|| \leq \varepsilon + CT(x,y). \tag{A.9}$$

Proof. **Proof of (1)**

Let $\Phi := [\Phi_1]_{\Phi_0}$ be the result of applying an iteration of the diffusion wavelet algorithm to $vol \cdot GD^{-1}$. So the columns of Φ (denoted $\{\phi_j\}_{1 \leq j \leq |S|}$) are an ε-span of the range of $volGD^{-1}$.

Let $\lambda_0, \lambda_1, ..., \lambda_{|S|}$ and $\zeta_0, \zeta_1, ..., \zeta_{|S|}$ denote the eigenvalues and eigenvectors of the unnormalized Green function G. Then we can define the subspace:

$$V_0 := \langle \{\zeta_\lambda : \lambda \geq \varepsilon\} \rangle.$$

The space spanned by the columns of Φ, $\tilde{V_0} := \langle \Phi \rangle$, is ε-close to V_0. In particular, from Definition 12 in [11], we have for all $\zeta \in V_0$:

$$||P_{\langle \Phi \rangle}\zeta - \zeta|| \leq \varepsilon,$$

where $P_{\langle \Phi \rangle}$ is the orthogonal projection onto $\tilde{V_0}$.

$$\begin{aligned} ||P_{\langle \Phi \rangle}(volGD^{-1})^{1/2}(\delta_x - \delta_y)|| &= ||P_{\langle \Phi \rangle}(volGD^{-1})^{1/2}(\delta_x - \delta_y) - (volGD^{-1})^{1/2}(\delta_x - \delta_y) \\ &\quad + (volGD^{-1})^{1/2}(\delta_x - \delta_y)|| \\ &\leq \varepsilon + ||(volGD^{-1})^{1/2}(\delta_x - \delta_y)|| \\ &\leq \varepsilon + \sqrt{CT(x,y)}. \end{aligned}$$

Also:

$$\begin{aligned} ||\Phi^T(\delta_x - \delta_y)|| &= ||\Phi^T(\delta_x - \delta_y) + (volGD^{-1})^{1/2}(\delta_x - \delta_y) - (volGD^{-1})^{1/2}(\delta_x - \delta_y)|| \\ &\geq -||\Phi^T(\delta_x - \delta_y) - (volGD^{-1})^{1/2}(\delta_x - \delta_y)|| + ||(volGD^{-1})^{1/2}(\delta_x - \delta_y)|| \\ &\geq -\varepsilon + \sqrt{CT(x,y)}. \end{aligned}$$

Therefore:

$$-\varepsilon + \sqrt{CT(x,y)} \leq ||\Phi^T(\delta_x - \delta_y)|| \leq \varepsilon + \sqrt{CT(x,y)}.$$

Proof of (2)

Note that for symmetric G, we have $\Theta^T\Theta \approx volGD^{-1}$. In general, using the first αN singular vectors of $volGD^{-1}$ leads to an approximation with error $\sigma_{\alpha N+1}$:

$$||(volGD^{-1})^{1/2} - (\Sigma^{1/2})^T U^T||_{2\to 2} = \sigma_{\alpha N+1}.$$

Hence

$$||\Theta^T(\delta_x - \delta_y) - (volGD^{-1})^{1/2}(\delta_x - \delta_y)||_2 = O(\sigma_{\alpha N+1}).$$

$$\begin{aligned}||\Theta^T(\delta_x - \delta_y)||_2 &\leq ||\Theta^T(\delta_x - \delta_y) - (volGD^{-1})^{1/2}(\delta_x - \delta_y)||_2 + ||(volGD^{-1})^{1/2}(\delta_x - \delta_y)||_2 \\ &\leq CT(x,y) + O(\sigma_{\alpha N+1}).\end{aligned}$$

Likewise,

$$\begin{aligned}||\Theta^T(\delta_x - \delta_y)|| &\geq -||\Theta^T(\delta_x - \delta_y) - (volGD^{-1})^{1/2}(\delta_x - \delta_y)|| + ||(volGD^{-1})^{1/2}(\delta_x - \delta_y)|| \\ &\geq -O(\sigma_{\alpha N+1}) + CT(x,y).\end{aligned}$$

Therefore:

$$-O(\sigma_{\alpha N+1}) + CT(x,y) \leq ||\Theta^T(\delta_x - \delta_y)|| \leq O(\sigma_{\alpha N+1}) + CT(x,y).$$

□

The above proofs assume that the embeddings Φ and Θ are obtained directly from the exact Green function G, but in the method proposed here, the Green function is approximated using compressed powers of T. This introduces additional error in the following way:

1. If the Green function is approximated using the diffusion wavelet algorithm with precision ε, then we have
$$\hat{G} = \prod_{k=0}^{K}(I + T_k),$$
where T_k approximates T^{2^k} up to ε. We have K such approximations, so $\hat{G}$ is a $K\varepsilon$-approximation of $\prod_{k=0}^{K}(I + T_k)$.

2. If the Green function is instead approximated using truncated SVD, where for $\alpha < 1$, $\alpha|S|$ singular values are retained at each step, then for each k, the approximation T_k introduces an error of magnitude $\sigma_{\alpha^k N+1}(T_{k-1})$. The total approximation error is then $\prod_{k=0}^{K}\sigma_{\alpha^k N+1}(T_k)$.

For either approximation, under the ideal assumptions on T from [11], there is some $\gamma < 1$ such that the rank of T^{2^j} up to precision ε is less than $\gamma\text{rank}_\varepsilon(T^{2^{j-1}})$, for all $j > 0$. In this case, the tail $\prod_K^\infty(I + T_k)$ is of low rank – specifically, $\text{rank}_\varepsilon(\prod_K^\infty(I + T_k) < \gamma\text{rank}_\varepsilon(T_K)$.

Theorem 3

The following theorem and its proof are based on the proof of Theorem 1 from [44].

Theorem. Suppose we have a kernel-based method (e.g. Q_1, Q_2 satisfying the properties in diffusion maps). Then the embeddings obtained from the diffusion wavelet scaling functions for $Q_2^{-1}Q_1$ at level j are equivalent, up to a rotation, to the p_j-dimensional embeddings obtained from the diffusion maps embedding, where p_j gives the number of scaling functions at scale j.

Proof. The diffusion maps algorithm embeds nodes by solving the eigenproblem

$$Q_2^{-1}Q_1 f = \lambda f,$$

and using $\Psi_t(x) := (\lambda_1^t \psi_1(x) \quad \lambda_2^t \psi_2(x) \cdots \lambda_{s(\delta,t)}^t \psi_{s(\delta,t)}(x))^T$ where

$$s = \max\{l \in \mathbb{N} \text{ such that } |\lambda_l|^t > \delta |\lambda_1|^t\}$$

and λ_k, ψ_k are the eigenvalues/eigenvectors of the transition matrix P. (With $1 = \lambda_0 > |\lambda_1| \geq \cdots$. Note that P and $Q_2^{-1}Q_1$ have the same eigenvectors, and the leading eigenvectors of one correspond to the last eigenvectors of the other.)

Let $j = \min\{k : \lambda_0^{t_k} > \varepsilon\}$. Take the first p_j eigenvectors of $Q_2^{-1}Q_1$, and let that be $V_{1:p_j}$. Let $[\phi_j]_{\phi_0}$ represent the scaling functions of $Q_2^{-1}Q_1$ at level j.

The scaling functions of $Q_2^{-1}Q_1$ at level j, by construction, give (an ε-approximation of) a localized base for $V_j = \langle \psi_\lambda : \lambda \in \sigma(T), \lambda^{t_j} \geq \varepsilon \rangle$. (This is the space corresponding to the p_j eigenvectors associated with the largest eigenvalues of P, or the eigenvectors associated to the p_j least eigenvalues of $Q_2^{-1}Q_1$).

Note that the columns of $V_{1:p_j}$ and $[\phi_j]_{\phi_0}$ span the same space. Hence

$$\begin{aligned} V_{1:p_j}^T V_{1:p_j} &= I = [\phi_j]_{\phi_0}^T [\phi_j]_{\phi_0}, \\ \Rightarrow V_{1:p_j} &= V_{1:p_j} V_{1:p_j}^T V_{1:p_j} = [\phi_j]_{\phi_0} ([\phi_j]_{\phi_0}^T V_{1:p_j}) \end{aligned}$$

Call this last matrix in parentheses Q. Then it can be shown that $QQ^T = Q^TQ = I$, so Q is a rotation matrix. Thus the p_j-dimensional diffusion map embeddings are equivalent up to a rotation to the jth-level scaling function-based embeddings.

$\square$

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] A. Ahmed et al. “Distributed Large-scale Natural Graph Factorization”. In: WWW 2013 - Proceedings of the 22nd International Conference on World Wide Web. May 2013, pp. 37–48. doi: 10.1145/2488388.2488393.

[2] R. Allen, P. Warren, and P. Wolde. “Sampling Rare Switching Events in Biochemical Networks”. In: Physical review letters 94 (Feb. 2005), p. 018104. doi: 10 . 1103 / PhysRevLett.94.018104.

[3] M. Belkin and P. Niyogi. “Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering”. In: Advances in Neural Information Processing System 14 (Apr. 2002). doi: 10.5555/2980539.2980616.

[4] P. Bolhuis et al. “Transition path sampling: throwing ropes over rough mountain passes in the dark”. In: Ann. Rev. Phys. Chem. 53 (2002), pp. 291–318. doi: 10.1146/annurev.physchem.53.082301.113146.

[5] H. Cai, V. Zheng, and K. Chang. “A Comprehensive Survey of Graph Embedding: Problems, Techniques and Applications”. In: IEEE Transactions on Knowledge and Data Engineering (Sept. 2017). doi: 10.1109/TKDE.2018.2807452.

[6] S. Cao, W. Lu, and Q. Xu. “GraRep”. In: CIKM ’15: Proceedings of the 24th ACM International on Conference on Information and Knowledge Management. Oct. 2015, pp. 891–900. doi: 10.1145/2806416.2806512.

[7] M. Chen, Q. Yang, and X. Tang. “Directed Graph Embedding”. In: IJCAI International Joint Conference on Artificial Intelligence. Jan. 2007, pp. 2707–2712.

[8] M. Chowdhury et al. “Can AI help in screening Viral and COVID-19 pneumonia?” In: IEEE Access 8 (2020), pp. 132665 –132676. doi: 10.1109/ACCESS.2020.3010287.

[9] F. Chung. “Laplacians and the Cheeger Inequality for Directed Graphs”. In: Annals of Combinatorics 9 (Apr. 2005), pp. 1–19. doi: 10.1007/s00026-005-0237-z.

[10] R. Coifman and S. Lafon. “Diffusion maps”. In: Applied and Computational Harmonic Analysis 21 (July 2006), pp. 5–30. doi: 10.1016/j.acha.2006.04.006.

[11] R. Coifman and M. Maggioni. “Diffusion wavelets”. In: Applied and Computational Harmonic Analysis 21 (July 2006), pp. 53–94. doi: 10.1016/j.acha.2006.04.004.

[12] T. Cragnolini et al. “Multifunctional energy landscape for a DNA G-quadruplex: An evolved molecular switch”. In: Journal of Chemical Physics 147 (Oct. 2017). doi: 10.1063/1.4997377.103

[13] I. Daubechies. “Orthonormal bases of compactly supported wavelets”. In: Comm. Pure Appl. Math. 41 (1988), pp. 909–996. doi: https://doi.org/10.1002/cpa.3160410705.

[14] J. Du and D. Liu. “Transitions states of stochastic chemical reaction networks”. In: Communications in Computational Physics 29.2 (2020), pp. 606–627. doi: 10.4208/ cicp.OA-2019-0014.

[15] C. Gardiner et al. “Handbook of Stochastic Methods for Physics, Chemistry and the Natural Sciences.” In: Biometrics 42 (Mar. 1986), p. 226. doi: 10.2307/2531274.

[16] D. Gillespie. “Approximate Accelerated Stochastic Simulation of Chemically Reacting Systems”. In: Journal of Chemical Physics 115 (July 2001), pp. 1716–1733. doi: 10. 1063/1.1378322.

[17] P. Goyal and E. Ferrara. “Graph Embedding Techniques, Applications, and Performance: A Survey”. In: Knowledge-Based Systems (May 2017). doi: 10.1016/j.knosys.2018. 03.022.

[18] A. Grover and J. Leskovec. “node2vec: Scalable Feature Learning for Networks”. In: vol. 2016. July 2016, pp. 855–864. doi: 10.1145/2939672.2939754.

[19] E. Hancock and H. Qiu. “Clustering and embedding using commute times”. In: IEEE Trans. Pattern Anal. Mach. Intell. 29 (2007), pp. 1873–1890.

[20] J. Leskovec, J. Kleinberg, and C. Faloutsos. “Graph Evolution: Densification and Shrinking Diameters”. In: ACM Transactions on Knowledge Discovery from Data (ACM TKDD) 1(1) (2007). doi: 10.1145/1217299.1217301.

[21] J. Leskovec and A. Krevl. SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data. June 2014.

[22] J. McAuley et al. Image-based Recommendations on Styles and Substitutes. 2015. doi: 10.1145/2766462.2767755. url: http://arxiv.org/abs/1506.04757.

[23] P. Mercurio and D. Liu. “Identifying transition states of chemical kinetic systems using network embedding techniques”. In: Mathematical Biosciences and Engineering 18.1 (2021), pp. 868–887. issn: 1551-0018. doi: 10.3934/mbe.2021046. url: https: //www.aimspress.com/article/doi/10.3934/mbe.2021046.

[24] P. Metzner, C. Schütte, and E. Vanden-Eijnden. “Transition Path Theory for Markov Jump Processes”. In: Mult. Mod. Sim. 7 (Nov. 2008), pp. 1192–1219. doi: 10.1137/ 070699500.

[25] P. Metzner, C. Schütte, and E. Vanden-Eijnden. “Illustration of transition path theory on a collection of simple examples”. In: The Journal of chemical physics 125 (Sept. 2006), p. 084110. doi: 10.1063/1.2335447.

[26] T. Mikolov et al. “Distributed Representations of Words and Phrases and their Compositionality”. In: Advances in Neural Information Processing Systems 26 (Oct. 2013). doi: 10.5555/2999792.2999959.

[27] M. Parrinello and A. Laio. “Escaping Free-Energy Minima”. In: Proc. Natl. Acad. Sci. U.S.A. 99 (Nov. 2002), p. 12562. doi: 10.1073/pnas.202427399. 104

[28] S. Pasquali. “HiRE-RNA: a high resolution coarse-grained energy model for RNA”. In: The journal of physical chemistry. B 114 (Sept. 2010), pp. 11957–66. doi: 10.1021/ jp102497y.

[29] B. Perozzi, R. Al-Rfou, and S. Skiena. “DeepWalk: Online Learning of Social Representations”. In: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (Mar. 2014). doi: 10.1145/2623330.2623732.

[30] L. Peshkin. MATLAB - GraphViz interface. May 2021. url: https://www.mathworks.com/matlabcentral/fileexchange/4518-matlab-graphviz-interface.

[31] E. Pettersen et al. “UCSF Chimera–a visualization system for exploratory research and analysis”. In: J Comput Chem 25(13) (Oct. 2004), pp. 1605–12. doi: 10.1002/jcc. 20084.

[32] T. Rahman et al. “Exploring the Effect of Image Enhancement Techniques on COVID-19 Detection using Chest X-ray Images.” In: Comput Biol Med (2021). doi: 10.1016/j.compbiomed.2021.104319.

[33] S. Roweis and L. Saul. “Nonlinear Dimensionality Reduction by Locally Linear Embedding”. In: Science (New York, N.Y.) 290 (Jan. 2001), pp. 2323–6. doi: 10.1126/ science.290.5500.2323.

[34] B. Rozemberczki. GraRep. 2019. url: https://github.com/benedekrozemberczki/ GraRep.

[35] F. Scarselli et al. “The Graph Neural Network Model”. In: IEEE Transactions on Neural Networks 20.1 (Jan. 2009), pp. 61–80. issn: 1941-0093. doi: 10.1109/TNN.2008. 2005605.

[36] P. Sen et al. Collective Classification in Network Data. 2008. doi: https://doi.org/10.1609/aimag.v29i3.2157.

[37] R. Srivastava, M. S. Peterson, and W. E. Bentley. “Stochastic kinetic analysis of the Escherichia coli stress circuit using Sigma-32-targeted antisense”. In: Biotechnology and Bioengineering 75.1 (2001), pp. 120–129. doi: 10.1002/bit.1171. eprint: https: / / onlinelibrary . wiley . com / doi / pdf / 10 . 1002 / bit . 1171. url: https : / / onlinelibrary.wiley.com/doi/abs/10.1002/bit.1171.

[38] R. Srivastava et al. “Stochastic vs. Deterministic Modeling of Intracellular Viral Kinetics”. In: Journal of theoretical biology 218 (Nov. 2002), pp. 309–21. doi: 10.1006/ jtbi.2002.3078.

[39] J. Tang et al. “LINE: Large-scale Information Network Embedding”. In: Line: Large- Scale Information Network Embedding (Mar. 2015). doi: 10.1145/2736277.2741093.

[40] D. J. Wales. “Exploring Energy Landscapes”. In: Annual Review of Physical Chemistry 69 (2018), pp. 401–25. doi: 10 . 1146 / annurev - physchem - 050317 - 021219. url: https://doi.org/10.1146/annurev-physchem-050317-021219.

[41] D. J.Wales. The Cambridge Energy Landscape Database. https://www-wales.ch.cam.ac.uk/CCD.html. url: https://www-wales.ch.cam.ac.uk/CCD.html. 105

[42] D. J. Wales and J. P. K. Doye. “Global Optimization by Basin-Hopping and the Lowest Energy Structures of Lennard-Jones Clusters Containing up to 110 Atoms”. In: The Journal of Physical Chemistry A 101.28 (July 1997), 5111–5116. issn: 1520-5215. doi: 10.1021/jp970984n. url: http://dx.doi.org/10.1021/jp970984n.

[43] B. Wang et al. “Network enhancement as a general method to denoise weighted biological networks”. In: Nature Communications (2018). doi: 10.1038/s41467-018-05469-x.

[44] C. Wang and S. Mahadevan. “Multiscale dimensionality reduction based on diffusion wavelets”. In: Tech. Rep., Univ. of Massachusetts (2009).

[45] J. Wang, K. Markert, and M. Everingham. “Learning models for object recognition from natural language descriptions”. In: Proceedings of British Machine Vision Conference. 2009. doi: 10.5244/C.23.2.

[46] M. Wang et al. Deep Graph Library: Towards Efficient and Scalable Deep Learning on Graphs. 2020. arXiv: 1909.01315 [cs.LG].

[47] E. Weinan and E. Vanden-Eijnden. “Towards a theory of transition paths”. In: J. Stat. Phys. 123 (2006), pp. 503–523. doi: 10.1007/s10955-005-9003-9.

[48] E. Wigner. “The transition state method”. In: Trans. Faraday Soc. 34 (1938), pp. 29–41. doi: 10.1007/978-3-642-59033-7_16.

[49] H. Yin et al. “Local Higher-order Graph Clustering”. In: In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2017. doi: 10.1145/3097983.3098069.

[50] W. Zachary. “An information flow model for conflict and fission in small groups”. In: Journal of anthropological research 33(4) (1977), 452–473. doi: 10.1086/jar.33.4. 3629752.

[51] D. Zhou, J. Huang, and B. Schoelkopf. “Learning from Labeled and Unlabeled Data on a Directed Graph”. In: Framework (Jan. 2005). doi: 10.1145/1102351.1102482. 106.