Title of dissertation: SPECTRAL ANALYSIS OF MARKOV JUMP PROCESSES WITH RARE TRANSITIONS: A GRAPH-ALGORITHMIC APPROACH

Tingyue Gan, Doctor of Philosophy, 2017

Dissertation directed by: Professor Maria Cameron
Department of Mathematics

Parameter-dependent Markov jump processes with exponentially small transition rates arise in modeling complex systems in physics, chemistry, and biology. Long-term dynamics of these processes are largely governed by the spectral properties of their generators. We propose a constructive graph-algorithmic approach to computing the asymptotic estimates of eigenvalues and eigenvectors of the generator matrix. In particular, we introduce the concepts of the hierarchy of Typical Transition Graphs (T-graphs) and the associated sequence of Characteristic Timescales. The hierarchy of T-graphs can be viewed as a unification of Wentzell’s hierarchy of optimal W-graphs and Friedlin’s hierarchy of Markov chains. T-graphs are capable of describing typical escapes from metastable classes as well as cyclic behaviors within metastable classes, for both reversible and irreversible processes, with or without symmetry. Moreover, the hierarchy of T-graphs can be used to construct asymptotic estimates of eigenvalues and eigenvectors simultaneously. We apply the proposed approach to investigate the biased random walk of a molecular motor and conduct zero-temperature asymptotic analysis of the LJ$_{75}$ network.
SPECTRAL ANALYSIS OF MARKOV JUMP PROCESSES WITH RARE TRANSITIONS: A GRAPH-ALGORITHMIC APPROACH

by

Tingyue Gan

Dissertation submitted to the Faculty of the Graduate School of the University of Maryland, College Park in partial fulfillment of the requirements for the degree of Doctor of Philosophy
2017

Advisory Committee:
Professor Maria Cameron, Chair/Advisor
Professor Alexander Barg
Professor Leonid Koralov
Professor Kasso Okoudjou
Professor Christopher Jarzynski
To my parents.
Acknowledgments

I consider myself very fortunate to have Professor Maria Cameron as my advisor. She has impressed me as a gritty pioneer woman scientist. During the past three years, she has offered me enormous patience and trust, and she has always encouraged me to explore my own ideas. I will forever be grateful for her guidance and kindness.

I would also like to express my gratitude to Professor Alexander Barg, Professor Matei Machedon, and Professor Konstantina Trivisa, for helping me through a very critical transition.

Last but not least, I would like to thank Professor Leonid Koralov, Professor Kasso Okoudjou, and Professor Christopher Jarzynski, for serving on my dissertation committee.
# Table of Contents

List of Figures vi

1 Introduction 1
   1.1 Metastable Markov Jump Processes .......................... 1
   1.2 Metastability via Spectral Analysis ........................ 4
   1.3 Main Results .............................................. 6

2 A Greedy Graph Algorithm for Computing Asymptotic Eigenvalues 11
   2.1 Optimal W-graphs ............................................ 12
   2.2 Asymptotic Estimates of Eigenvalues ........................ 13
   2.3 Weak Nested Property of Optimal W-graphs .................. 14
   2.4 Metastability and Optimal W-graphs ......................... 16
   2.5 The Construction of Optimal W-graphs ....................... 18
      2.5.1 The Ideas .............................................. 18
      2.5.2 The Recursive Structure of OptWgraphs() .............. 21
      2.5.3 The Functions Contract_W() and Expand_W() ........... 22
      2.5.4 An Illustrative Example ............................... 27

3 T-graphs and Asymptotic Eigenvectors 30
   3.1 The Hierarchy of Typical Transition Graphs and Characteristic Timescales 32
      3.1.1 The Recursive Structure of Tgraphs() ................... 32
      3.1.2 The Functions Contract_T() and Expand_T() .......... 33
      3.1.3 An Illustrative example .............................. 37
   3.2 Communicating Classes in T-graphs and Freidlin’s Hierarchy of Markov Chains ......................... 40
   3.3 Metastability and T-graphs .................................. 44
   3.4 Asymptotic Estimates of Eigenvectors ........................ 47
   3.5 A Non-Recursive Implementation Scheme for Constructing T-graphs 48

4 Applications 51
   4.1 Biasing the Random Walk of a Biomolecular Motor ............. 51
   4.2 Asymptotic Analysis of The LJ75 Network .................... 57
5 Conclusion
A Proof of Theorem 2.2.3
B Proof of Theorem 2.3.1
C Proof of Theorem 2.5.2
D Proof of Theorem 3.4.1
E Pseudocodes of a Non-Recursive Implementation of Algorithm 2
Bibliography
List of Figures

2.1 An example of W-graph and optimal W-graph . . . . . . . . . . . . . 13
2.2 An illustration of the weak nested property of optimal W-graphs . . 16
2.3 An example illustrating that unique optimal W-graphs does not imply
no symmetry . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 22
2.4 The flowchart of Algorithm 1 . . . . . . . . . . . . . . . . . . . . . 24
2.5 An illustrative example of Algorithm 1 . . . . . . . . . . . . . . . . 29

3.1 The flowchart of Algorithm 2 . . . . . . . . . . . . . . . . . . . . . 35
3.2 An illustrative example of Algorithm 2 . . . . . . . . . . . . . . . . 41
3.3 An example of Freidlin’s hierarchy of Markov Chains . . . . . . . . 44
3.4 Freidlin’s hierarchy of Markov Chains for $G$ in Figure 3.2 . . . . . 45

4.1 The time-dependent free energy landscape of Kinesin . . . . . . . . . 53
4.2 The hierarchy of T-graphs of the eight-state random walk model of
the Kinesin motor when $\zeta \in (6, 9.5)$. . . . . . . . . . . . . . . . . 55
4.3 The T-graphs on the timescales when the Kinesin motor is most likely
to accomplish the first forward motion for various ranges of values of $\zeta$. 56
4.4 The disconnectivity graph of the energy landscape of LJ$_{75}$. . . . . 58
4.5 The exponential factors $\Delta_k$ for the LJ$_{75}$ network. . . . . . . . 60
4.6 The energy along the asymptotic zero-temperature path connecting
the two lowest minima 1 and 92 of LJ$_{75}$ . . . . . . . . . . . . . . . . . 61
Chapter 1: Introduction

1.1 Metastable Markov Jump Processes

Metastability is a ubiquitous phenomenon that arises in nature and society. For example, in biochemistry, a macromolecule can change its shape in response to changes in its environment (temperature, PH, etc). Each possible shape is called a conformation. A macromolecule can stay in one conformation for a relatively long period of time before it abruptly changes to another conformation. Other complex systems manifesting metastability include lattice spin models at low temperature, genetic population dynamics [10], and financial crises.

A popular choice of mathematical model for studying the long-term behavior of metastable systems is a parameter-dependent Markov jump process with a finite state space and exponentially distributed holding times. The dynamics of the process is described by the generator matrix $L$. Each off-diagonal entry $L_{ij}$ is the transition rate from state $i$ to state $j$, which often takes the form of

$$L_{ij} \approx \exp \left( -U_{ij}/\varepsilon \right), \text{ i.e. } \lim_{\varepsilon \to 0} \varepsilon \ln L_{ij} = -U_{ij}, \quad (1.1)$$

where $U_{ij} \geq 0$ is the exponential factor and $\varepsilon > 0$ is a small parameter. Transitions are rare when $\varepsilon$ is very small. The diagonal entries are defined as $L_{ii} = -\sum_{j \neq i} L_{ij}$.
so that the generator matrix \( L \) has zero row sums. The nonnegative numbers \(-L_{ii}\) are the escape rates from states \( i \).

Long-term behavior of stochastic processes with rare transitions has been attracting attention of mathematicians for the last fifty years [2, 17, 22, 23]. Freidlin and Wentzell [17] developed the Large Deviation Theory in their study of random perturbations of dynamical systems. They showed that the long-term behavior of a diffusion processes \( X_t \) generated by the SDE

\[
dX_t = b(X_t)dt + \sqrt{2\varepsilon}dW_t
\]

(1.2)
can be effectively modeled by a Markov chain whose states correspond to the stable attractors of \( \dot{x}_t = b(x_t) \).

To estimate the exponential factor \( U_{ij} \), they described the large deviations of \( X_t \) from the deterministic trajectory \( \dot{x}_t = b(x_t) \) by introducing the action functional

\[
I_T(\phi) = \frac{1}{2} \int_0^T \| \dot{\phi}(t) - b(\phi(t)) \|^2 dt
\]

(1.3)
defined on the space of absolute continuous paths \( \phi \), and the quasi-potential

\[
U_{ij} = \inf \{ I_T(\phi) \mid \phi(0) \in K_i, \ \phi(T) \in K_j, \ T > 0 \},
\]

(1.4)
which characterizes the difficulty of the passage from attractor \( i \) to attractor \( j \). Here \( K_i \) and \( K_j \) are two compact sets corresponding to the two distinct attractors.

A Markov jump process with pairwise transition rates of the form of Eq. (1.1) can be represented, up to exponential order, by a weighted directed graph \( G(S, A, U) \), where the set of vertices \( S \) is the set of states, the set of arcs \( A \) includes only those arcs \( (i \rightarrow j) \) such that \( L_{ij} > 0 \), and \( U = \{ U_{ij} \mid (i \rightarrow j) \in A \} \) is the set of
arc weights representing exponential factors of the transition rates. An arc \((i \to j)\) is a directed edge going from vertex \(i\) to vertex \(j\), and the vertices \(i\) and \(j\) are called \textit{tail} and \textit{head}, respectively.

The following is assumed throughout the dissertation.

**Assumption 1.1.1.** All Markov jump processes under consideration have finite states and are irreducible.

Irreducibility means that it is possible to get to any state from any state. A state \(i\) is said to \textit{communicate} with state \(j\) if it is possible to get to \(i\) from \(j\) and vice versa. A \textit{communicating class} \(C\) is a maximal set of states such that every pair of states in \(C\) communicates with each other. A communicating class is \textit{closed} if it is impossible to leave the class, otherwise it is \textit{open}. In terms of the graph \(G\) representing a Markov jump process, irreducibility is equivalent to \(G\) being a single communicating class, i.e. strongly connected. Assumption 1.1.1 guarantees the existence and uniqueness of an invariant distribution \(\pi\) satisfying \(\pi^T L = 0\), \(\pi_i > 0\), and \(\sum_{i \in S} \pi_i = 1\).

We emphasize that reversibility is not assumed. In other words, the process is not required to satisfy the \textit{detailed balance} equation \(\pi_i L_{ij} = \pi_j L_{ji}\). This is desirable since many real-world processes are best described by an irreversible model. In some scenarios, even if the model is reversible, if we are interested in the long-term dynamics of a very metastable model, it is often hard to sample from simulations. Adding external forces might help accelerate the sampling, but it also renders the model irreversible.
1.2 Metastability via Spectral Analysis

It is well known that long-term dynamics of a metastable system is largely governed by the spectral properties of its generator. The case of reversible Markov processes has been well understood over the last decades. The key is the self-adjointness of the generator. Using the classic potential theory as a tool, Bovier et al. [2–5] derived sharp estimates for low-lying spectra of reversible Markov processes, defined a hierarchy of metastable sets, as well as identified the link between eigenvalues and expected exit times. As is also known, the slowest relaxation processes are described by the dominant eigenvalues and eigenvectors of the generator, and the Markov State Model offers an efficient tool for their computation [23]. Using the minimum spanning tree, Cameron [7] proposed an efficient algorithm to compute asymptotic estimates of eigenvalues and eigenvectors of the generator, which is suitable for large and complex networks. However, all these methods fall short in the case of irreversible Markov processes. The goal of this dissertation is to develop a program capable of analyzing both reversible and irreversible Markov jump processes, at both qualitative and quantitative levels.

We begin by discussing some fundamental spectral properties of the generator matrix $L$ with transition rates given by Eq. (1.1). The zero row sum property of the generator matrix $L$ indicates that $\lambda_0 = 0$ is an eigenvalue, and the corresponding right eigenvector is $\varphi_0 = 1$, a column vector with every entry equals to 1. Since $\pi^T L = 0$, the corresponding left eigenvector is $\psi_0 = \pi$, the invariant distribution. Because $\pi$ is unique, $\lambda_0 = 0$ is a simple eigenvalue.
Let $\lambda_1, \lambda_2, \ldots, \lambda_{n-1}$ be the nonzero eigenvalues of $-L$, indexed in increasing order of their magnitudes. Due to the diagonal dominance of $-L$ and positivity of its diagonal entries, all nonzero eigenvalues have positive real parts, i.e., $\text{Re}(\lambda_k) > 0$ for $k = 1, 2, \ldots, n - 1$. This can be shown by applying Gershgorin’s circle theorem. It is clear from the form of $L_{ij}$ (Eq. 1.1) that all nonzero $\text{Re}(\lambda_k)$ are exponentially small and converge to zero as the parameter $\varepsilon$ goes to zero.

In the case where all $\lambda_k$’s are real and distinct, one can write the probability distribution $p(t)$, governed by the Fokker-Planck equation $dp/dt = pL^T$, in the following form

$$p(t) = \pi + \sum_{k=1}^{n-1} e^{-\lambda_k t}(p(0)^T \varphi_k)\psi_k.$$  \hspace{1cm} (1.5)

Keep in mind that the eigenvalues $\lambda_k$, corresponding right and left eigenvectors $\varphi_k$ and $\psi_k$, and the invariant distribution $\pi$ are all functions of the small parameter $\varepsilon$. The inverses of the nonzero eigenvalues separate the time horizon into several time spans, such that the asymptotic behavior of the process is qualitatively different during each time span. Specifically, let $t$ be a timescale satisfying $\frac{1}{\tilde{k}+1} \ll t \ll \frac{1}{k}$, or equivalently, $\lim_{\varepsilon \to 0} \varepsilon \ln \frac{1}{\tilde{k}+1} < \lim_{\varepsilon \to 0} \varepsilon \ln t < \lim_{\varepsilon \to 0} \varepsilon \ln \frac{1}{k}$, for some $\tilde{k}$ between 1 and $n - 2$, then

$$\lim_{\varepsilon \to 0} \lambda_k t = \begin{cases} 0, & 1 \leq k \leq \tilde{k}, \\ \infty, & \tilde{k} + 1 \leq k \leq n - 1. \end{cases}$$  \hspace{1cm} (1.6)

Consequently,

$$\lim_{\varepsilon \to 0} p(t) = \lim_{\varepsilon \to 0} \left( \pi + \sum_{k=1}^{\tilde{k}} (p(0)^T \varphi_k)\psi_k \right).$$  \hspace{1cm} (1.7)
That is to say, the $k$th eigen-component of the probability distribution $p(t)$ will remain significant until the timescale surpasses $\lambda_k^{-1}$.

In many real-world applications, direct calculations of the eigenvalues and eigenvectors of the generator matrices are often forbidden, due to the curse of dimensionality and issues related to floating-point arithmetic when $\varepsilon$ is very small. Yet it is of considerable practical interest to be able to estimate the eigenvalues and eigenvectors as precisely as possible since these are the quantities controlling metastable behavior. For reversible Markov processes, using the minimum spanning tree, [7] developed a graph-algorithmic approach to computing the asymptotic estimates of eigenvalues and eigenvectors of the generator matrices starting from the low-lying group. This dissertation extends substantially, the graph-algorithmic approach in [7] to tackle both reversible and irreversible Markov processes, which has its root in the fundamental results established by Freidlin and Wentzell, in particular, Wentzell’s asymptotic estimates of eigenvalues using so-called optimal W-graphs and the so-called Freidlin’s hierarchy of Markov chains.

1.3 Main Results

We highlight the main results. For more details and further references, see the respective sections. Parts of these results are published in [8, 19].

In Chapter 2, under the assumption that all optimal W-graphs are unique (Assumption 2.2.2), we derive the following results:

(1) In section 2.2, we obtain refined asymptotic estimates of eigenvalues includ-
ing pre-factors when the transition rates are of the specific form: \( L_{ij} = \kappa_{ij} \exp(-U_{ij}/\varepsilon) \). This refinement is based on Wenzell’s incisive expressions for asymptotic estimates of eigenvalues in terms of optimal W-graphs when the transition rates \( L_{ij} \) are of the order of \( \exp(-U_{ij}/\varepsilon) \) (Eq. (1.1)).

2) In section 2.3, we establish a so-called weak nested property of optimal W-graphs, which describes a hierarchical structure of the collection of optimal W-graphs. It says that when one connected component of an optimal W-graph collapses into another one of its connected components in a certain manner, the resulting graph is the subsequent optimal W-graph having one sink less. This weak nested property is a red thread that connects most of our results.

3) In section 2.4, we discuss how the weak nested property of optimal W-graphs can help us visualize the typical evolution of a metastable system across a sequence of exponentially increasing timescales. In particular, how the sets of sinks \( W_k^* \) of optimal W-graphs can be seen as the sets of metastable states for the time spans \((\lambda_k^{-1}, \lambda_{k-1}^{-1})\).

4) In section 2.5, we propose a greedy/dynamic programming algorithm (Algorithm 1) for constructing the hierarchy of optimal W-graphs as well as extracting the exponential factors of the asymptotic estimates of eigenvalues. This algorithm is conceptually recursive, it exploits the weak nested property of optimal W-graphs, and it is inspired by the ideas of contraction and expansion of cycles from Chu-Liu/Edmond’s algorithm for finding optimal branching.
The correctness of this algorithm is proved under a no-symmetry assumption (Assumption 2.5.1).

To handle the case with symmetry, in Chapter 3, we introduce the concepts of the hierarchy of Typical Transition Graphs (T-graphs) and the associated sequence of Characteristic Timescales.

(5) In section 3.1, we define the hierarchy of T-graphs and the associated characteristic timescales inductively via an Algorithm 2, which is a modification of Algorithm 1 and is recursive as well. T-graphs record all typical transitions as well as their effective transition rates up to the associated characteristic timescales, and this hierarchy is strongly nested. T-graphs differ from optimal W-graphs in two critical ways, they contain communicating classes and vertices in T-graphs also carry weights.

(6) In section 3.2, we discuss how the collection of closed communicating classes encountered by Algorithm 2 during the construction of T-graphs coincides with Freidlin’s hierarchy of Markov chains. Hence, the hierarchy of T-graphs unifies the hierarchy of optimal W-graphs and Freidlin’s hierarchy of Markov chains.

(7) In section 3.3, we discuss why the closed communicating classes in each T-graph can be viewed as metastable classes between two characteristic timescales, and how we can calculate estimates of metastable distributions on these closed communicating classes using vertex weights of the respective T-graphs.
(8) In section 3.4, under the assumption that all optimal W-graphs are unique, we construct asymptotic estimates of eigenvectors using T-graphs, where we once again harness the power of the weak nested property. The asymptotic estimates of left eigenvectors are expressed in terms of the metastable distributions on the collapsing and absorbing closed communicating classes, while the asymptotic estimates of right eigenvectors are expressed as indicator functions of the collapsing connected components.

(9) In section 3.5, we propose a non-recursive implementation scheme for constructing the hierarchy of T-graphs (Algorithm 2) which is object-oriented.

Chapter 4 is devoted to applications.

(10) In section 4.1, we apply the concept of T-graphs to inspect how an external control parameter can bias the random walk of a Kinesin motor and predict the timescales when the motor is most likely to accomplish the first forward motion for various ranges of values of the control parameter.

(11) In section 4.2, we apply Algorithm 1 (optimal W-graphs) to conduct zero-temperature asymptotic analysis of the stochastic network representing the energy landscape of Lennard-Jones cluster of 75 atoms (LJ\textsubscript{75} network). The energy landscape of LJ\textsubscript{75} has a double-funnel structure, and the process of physical interest is the transition from the second lowest minimum to the global minimum. Algorithm 1 allows us to extract the exponential factor of the associated exit rate as well as predict the most likely zero-temperature
transition path by tracing the unique directed path from the second lowest minimum to the global minimum in the associated optimal W-graph.
Chapter 2: A Greedy Graph Algorithm for Computing Asymptotic Eigenvalues

Optimal W-graphs were introduced by Wentzell [17, 24] when deriving asymptotic estimates of eigenvalues of the generator matrix $L$. The significance of Wentzell’s result is that it translates the problem of estimating eigenvalues to that of finding the hierarchy of optimal W-graphs, which is nevertheless a rather challenging combinatoric optimization problem. Under the assumption that all optimal W-graphs are unique, in section 2.2, we obtain refined asymptotic estimates of eigenvalues when pre-factors of transition rates are known; in section 2.3, we establish the weak nested property of optimal W-graphs; and in section 2.4, we discuss how the weak nested property can be used to describe escape behaviors. Illuminated by the physical insights behind the weak nested property, and further inspired by the ideas of contraction and expansion of cycles, in section 2.5, we propose a greedy/dynamical programming algorithm for constructing optimal W-graphs and extracting exponential factors of the asymptotic estimates of eigenvalues.
2.1 Optimal W-graphs

Let $G(S, A, U)$ be the weighted strongly connected directed graph associated with the generator matrix $L$. A spanning subgraph of $G$ includes a subset of arcs of $G$ and all its vertices. Therefore, we can identify a spanning subgraph with its set of arcs.

**Definition 2.1.1. (W-graph)** A W-graph with $k$ sinks is a spanning subgraph of $G$ such that

(i) each vertex either emits exactly one arc or none, and there are $k$ vertices (sinks) emitting none;

(ii) it contains no cycles.

For all $k = 1, 2, \cdots, n$, the strong connectivity of $G$ ensures the existence of a W-graph with $k$ sinks (note that a W-graph with $n$ sinks is a spanning subgraph with no arcs). A cycle is oriented with each of its vertices emitting as well as receiving exactly one arc. A W-graph with $k$ sinks is essentially a forest with $k$ in-trees (thus $n - k$ arcs). In each in-tree, there is a unique directed path leading from any non-root vertex to its root (sink). Denote by $G(k)$ the collection of all possible W-graphs with $k$ sinks.

**Definition 2.1.2. (Optimal W-graph)** $g^*_k$ is an optimal W-graph with $k$ sinks if and only if

$$g^*_k = \arg \min_{g \in G(k)} V(g)$$

(2.1)
where \( \mathcal{V}(g) = \sum_{(i \to j) \in g} U_{ij}, \ k = 1, 2, \cdots, n. \)

We emphasize that Eq. (2.1) minimizes over all feasible choices of sinks and arcs. Figure 2.1 shows a simple example of W-graph and optimal W-graph.

![Figure 2.1: Left: the underlying graph \( G \). Middle: a W-graph with two sinks, 2 and 3, its total weight is 7. Right: the unique optimal W-graph with two sinks, 3 and 4, its total weight is 4. Blue and yellow vertices correspond to sinks and non-sinks, respectively.](image)

### 2.2 Asymptotic Estimates of Eigenvalues

In 1972, Wentzell established the following asymptotic estimates of eigenvalues in terms of the optimal W-graphs [24].

**Theorem 2.2.1. (Wentzell, 1972)** Let \( 0, -\lambda_1, -\lambda_2, \cdots, -\lambda_{n-1} \) be the eigenvalues of a generator matrix \( L \) with off-diagonal entries of the order of \( \exp(-U_{ij}/\varepsilon) \), indexed so that \( 0 < |\lambda_1| \leq |\lambda_2| \leq \cdots \leq |\lambda_{n-1}| \). Then as \( \varepsilon \to 0 \),

\[
\text{Re}(\lambda_k) \asymp \exp(-\Delta_k/\varepsilon),
\]

where \( \Delta_k = \mathcal{V}(g^*_k) - \mathcal{V}(g^*_{k+1}). \)

Theorem 2.2.1 implies that \( \Delta_1 \geq \Delta_2 \geq \cdots \geq \Delta_{n-1}. \)

Assumption 2.2.2 below enables us to derive a variety of fundamental results.
Assumption 2.2.2. All optimal Wgraphs $g_k^\ast$ are unique.

Under Assumption 2.2.2, $\Delta_k$’s are strictly decreasing, i.e., $\Delta_1 > \Delta_2 > \cdots > \Delta_{n-1}$. Hence, for $\varepsilon$ sufficiently small, all $\lambda_k$’s are real and distinct. If in addition, pre-factors of transition rates are known, specifically, if $L_{ij} = \kappa_{ij} \exp(-U_{ij}/\varepsilon)$, then refined estimates of eigenvalues can be obtained.

**Theorem 2.2.3.** Under Assumption 2.2.2, let $0 > -\lambda_1 > -\lambda_2 > \cdots > -\lambda_{n-1}$ be the eigenvalues of a generator matrix $L$ with off-diagonal entries equal to $\kappa_{ij} \exp(-U_{ij}/\varepsilon)$. Then there exits $\rho > 0$ such that,

$$
\lambda_k = \alpha_k \exp(-\Delta_k/\varepsilon) \left(1 + o(\exp(-\rho/\varepsilon))\right),
$$

where $\Delta_k = \mathcal{V}(g_k^\ast) - \mathcal{V}(g_{k+1}^\ast)$ and $\alpha_k = \frac{\prod_{(i\rightarrow j) \in g_k^\ast} \kappa_{ij}}{\prod_{(i\rightarrow j) \in g_{k+1}^\ast} \kappa_{ij}}$.

The proof of Theorem 2.2.3 is provided in Appendix A. The key is the connection between the coefficients of the characteristic polynomial of $L$ and the collection of W-graphs with the corresponding number of sinks.

2.3 Weak Nested Property of Optimal W-graphs

In this section, we will establish a weak nested property of optimal W-graphs under Assumption 2.2.2. This property is pivotal, not only does it serve as a stepping stone in our algorithm design, but it also facilitates deeper understandings of metastable behavior.

An in-tree of a W-graph $g_k$ is an induced subgraph of $g_k$. It includes a subset of vertices of $g_k$ and all the arcs of $g_k$ whose endpoints belong to the vertex subset. Therefore, we can identify an in-tree of a W-graph with its set of vertices.
For a subset of vertices $X \subset S$, denote by $A(g; X)$ the subset of arcs of $g$ with tails in $X$. Denote by $W_k^*$ the set of sinks of the optimal W-graph $g_k^*$.

**Theorem 2.3.1.** Under Assumption 2.2.2, the collection of optimal W-graphs $\{g_k^*\}_{k=1}^n$ is a hierarchy satisfying the following properties:

(i) There is a unique in-tree $S_k^+$ of $g_k^*_{k+1}$ whose sink $s_k^+$ is not a sink of $g_k^*$, i.e.,
$$W_k^* = W_{k+1}^* \setminus \{s_k^+\}.$$  

(ii) $A(g_k^*; S \setminus S_k^+) = A(g_{k+1}^*; S \setminus S_k^+)$, i.e., all the arcs of $g_k^*$ with tails not in $S_k^+$ are inherited from $g_{k+1}^*$. However, $A(g_k^*; S_k^+ \setminus \{s_k^+\})$ and $A(g_{k+1}^*; S_k^+ \setminus \{s_k^+\})$ do not necessarily coincide.

(iii) There is a single arc $(p_k \rightarrow q_k)$ in $g_k^*$ with tail $p_k$ in $S_k^+$ and head $q_k$ in another in-tree $S_k^-$ of $g_{k+1}^*$.

Theorem 2.3.1 is illustrated in Figure 2.2. The optimal W-graph $g_k^*$ is obtained from $g_{k+1}^*$ by (1) connecting $S_k^+$ and $S_k^-$ of $g_{k+1}^*$ via an arc $(p_k \rightarrow q_k)$, and (2) possibly rearranging some arcs within $S_k^+$. We say that the sink $s_k^+$ of $S_k^+$ collapses, all flow there is absorbed by $S_k^-$ and gravitates to the sink $s_k^-$. The arc $(p_k \rightarrow q_k)$ is like a bottleneck.

The proof of Theorem 2.3.1 is provided in Appendix B. The main technique involved is the swapping of subsets of arcs.
Figure 2.2: An illustration of the weak nested property of optimal W-graphs. The ovals symbolize in-trees of the optimal W-graphs. Arcs in the gray components remain the same. Blue and yellow vertices correspond to sinks and non-sinks, respectively.

2.4 Metastability and Optimal W-graphs

When all optimal W-graphs are unique (Assumption 2.2.2), the hierarchy of sink sets $W_k^*$’s can be seen as a hierarchy of sets of metastable states. Specifically, for the time span $(\lambda_k^{-1}, \lambda_{k+1}^{-1})$, the set of sinks $W_{k+1}^*$ of $g_{k+1}^*$ are metastable, in the sense that if a process starts from a state belonging to some in-tree of $g_{k+1}^*$, then with probability tends to 1 as $\varepsilon \to 0$, the process will be found at the sink of that in-tree on any timescale $t \in (\lambda_{k+1}^{-1}, \lambda_k^{-1})$.

As timescale reaches the order of $\lambda_k^{-1}$, the least stable sink $s_k^+$ in $W_{k+1}^*$ collapses and starts emitting flow. Almost instantaneously (compared with the timescale of the order of $\lambda_k^{-1}$), the flow is absorbed by the in-tree $S_k^-$ and is passed to its sink $s_k^-$. Any timescale $t \in (\lambda_k^{-1}, \lambda_{k-1}^{-1})$, however, is not large enough for any sink in
$W_{k+1}^*$ other than $s_k^+$ to collapse. During the time span $(\lambda_k^{-1}, \lambda_{k-1}^{-1})$, a process starting from any in-tree of $g_{k+1}^*$ other than $S_k^*$ will be stuck in that in-tree, and will spend most of its time at the sink. Hence, $W_k^* = W_{k+1}^* \setminus \{s_k^+\}$ becomes the set of metastable states for the subsequent time span $(\lambda_k^{-1}, \lambda_{k-1}^{-1})$.

To further quantify such metastable behavior, we invoke Lemma 3.3 and Lemma 3.4 from Chapter 6 of [17]. Let $\mathbb{P}_x$ be the law of the Markov jump process $X_t$ starting from $x$. The first passage time of a subset $W$ of $S$ is $\tau_W = \inf\{t \geq 0 \mid X_t \in W\}$. Denote by $\mathcal{G}(W)$ the collection of W-graphs with set of sinks $W$. For $x \notin W$ and $y \in W$, let $\mathcal{G}_{x,y}(W)$ denote the subset of $\mathcal{G}(W)$ consisting of those with a path leading from $x$ to $y$.

For a process starting from $x \notin W_k^*$, the probability that at the first passage time of $W_k^*$ it hits $y \in W_k^*$ is estimated by Eq. (2.4).

$$\mathbb{P}_x(X_{\tau_{W_k^*}} = y) \asymp \frac{\sum_{g \in \mathcal{G}_{x,y}(W_k^*)} \Pi(g)}{\sum_{g \in \mathcal{G}(W_k^*)} \Pi(g)} \asymp \frac{\exp(-\min\{g \in \mathcal{G}_{x,y}(W_k^*)\} V(g)/\varepsilon)}{\exp(-V(g_k^*)/\varepsilon)}, \quad (2.4)$$

where $\Pi(g) = \prod_{(i \to j) \in g} L_{ij}$. Hence,

$$\lim_{\varepsilon \to 0} \mathbb{P}_x(X_{\tau_{W_k^*}} = y) = \begin{cases} 1, & \text{if } x \text{ leads to } y \text{ in } g_k^*, \\ 0, & \text{otherwise}. \end{cases} \quad (2.5)$$

The expected first passage time of $W_k^*$ is estimated by Eq. (2.6).

$$\mathbb{E}_x[\tau_{W_k^*}] \asymp \frac{\sum_{g \in \mathcal{G}(W_k^*) \cup \{x\}} \Pi(g) + \sum_{z \notin W_k^*, z \neq x} \sum_{g \in \mathcal{G}_{x,z}(W_k^* \cup \{z\})} \Pi(g)}{\sum_{g \in \mathcal{G}(W_k^*)} \Pi(g)}. \quad (2.6)$$

Since the numerator of the RHS of Eq. (2.6) sums over a subset of W-graphs with $k+1$ sinks, $\mathbb{E}_x[\tau_{W_k^*}] \leq \frac{\exp(-V(g_{k+1}^*)/\varepsilon)}{\exp(-V(g_k^*)/\varepsilon)} \asymp \lambda_k^{-1}$. In fact, one can deduce from the
weak nested property of optimal W-graphs that

\[
\mathbb{E}_x[\tau_{W_k^*}] \begin{cases} 
\asymp \lambda_k^{-1}, & \text{if } x \in S_k^+ \\
\ll \lambda_k^{-1}, & \text{if } x \not\in S_k^+.
\end{cases} \quad (2.7)
\]

2.5 The Construction of Optimal W-graphs

2.5.1 The Ideas

We know from last section that under Assumption 2.2.2, a process starting from the in-tree \(S_k^+\) spends most of its time at the sink \(s_k^+\) prior to its exit from \(S_k^+\) on the timescale \(t \asymp \lambda_k^{-1}\). Let \(t_k^+\) denote the expected holding time at the sink \(s_k^+\), depending on whether \(t_k^+ \asymp \lambda_k^{-1}\) or \(t_k^+ \ll \lambda_k^{-1}\), the process will most likely behave quite differently prior to its exit.

(a) \(t_k^+ \asymp \lambda_k^{-1}\). The process first appears to be frozen at \(s_k^+\). As timescale reaches the order of \(\lambda_k^{-1}\), it suddenly takes a direct jump from \(s_k^+\) to some \(q_k\) in \(S_k^-\). In this case, \(p_k = s_k^+\) and \(t_k^+ \asymp \lambda_k^{-1} \asymp L_{s_k^+q_k}^{-1}\). The transition rate \(L_{s_k^+q_k}\) is the highest among all possible exits from \(s_k^+\), or equivalently, \((s_k^+ \rightarrow q_k)\) has the smallest weight \(U_{s_k^+q_k}\) among all outgoing arcs from \(s_k^+\).

(b) \(t_k^+ \ll \lambda_k^{-1}\). The process first performs rotations within \(S_k^+\) which pass the sink \(s_k^+\). Assume for simplicity, all rotations are along a single cycle \(c\). As timescale reaches the order of \(\lambda_k^{-1}\), it suddenly breaks through the cycle \(c\) via a jump \(p_k \rightarrow q_k\), where \(q_k\) is in \(S_k^-\) and \(p_k\) can be any vertex in the cycle \(c\), not necessarily \(s_k^+\). The cycle \(c\) can be viewed as a super-state in \(S_k^+\), and the
process exits $S_k^+$ through this super-state.

Guided by the physical insights above, we exploit the weak nested property and make the following educated guess regarding the structure evolution of optimal W-graphs.

(a') A direct jump $s_k^+ \rightarrow q_k$. In this case, $g_k^*$ is simply obtained by adding the arc $(s_k^+ \rightarrow q_k)$ to $g_{k+1}^*$.

(b') Rotations along a cycle $c$ prior to an exit jump $p_k \rightarrow q_k$. In this case, (1) the exit arc $(p_k \rightarrow q_k)$ shall be in $g_k^*$; (2) except for the arc with tail $p_k$, all arcs in the cycle $c$ shall be in $g_k^*$; (3) arcs of $g_{k+1}^*$ whose tails do not belong to the cycle $c$ shall all be inherited by $g_k^*$.

Starting from the optimal W-graph $g_n^*$ which consists of no arcs, we will build the optimal W-graphs one upon another, in the order of $g_{n-1}^*, \cdots, g_2^*, g_1^*$, by means of adding and rearranging arcs. We will identify a W-graph with its set of arcs since a W-graph is a spanning subgraph of $G$.

Let $A_0$ denote the subset of $A$ consisting of all the minimum weight outgoing (min-outgoing) arcs from each vertex of $G$. Assume for simplicity that every vertex has a unique min-outgoing arc and no two min-outgoing arcs are of the same weight, then $A_0$ contains exactly $n$ arcs. Let $A_0(1 : l)$ denote the subset of $A_0$ consisting of the first $l$ minimum weight arcs. (Note $A_0(1 : 0) = \emptyset$.)

We start with two sets $B = \emptyset$ and $B_0 = A_0$, one at a time, we transfer the minimum weight arc (min-arc) in $B_0$ to $B$, stop if either a cycle $c$ is formed in $B$ or there is only one arc left in $B_0$. Suppose we have made $n - k_1$ transfers right before
we stop. It is easy to see that \( g_k^* = A_0(1 : n - k) \) for \( k = k_1, \ldots, n - 1 \). If \( k_1 = 1 \), we are done, all optimal W-graphs are found trivially. However, if \( k_1 > 1 \), we have encountered a cycle \( c \), and the optimal W-graphs with sinks fewer than \( k_1 \) are yet to be found.

To proceed, we borrow the ideas of contraction and expansion of cycles from Chu-Liu/Edmonds’ algorithm [12] for finding optimal branching, the directed analog of the minimum spanning tree problem. Chu-Liu/Edmonds’ algorithm can be used to find the optimal W-graph with one sink. However, we want the full hierarchy of optimal W-graphs, in other words, we need to solve a sequence of combinatoric optimization problems. The beauty here is that when the ideas of contraction and expansion fuse with the weak nested property, constructing the hierarchy of optimal W-graphs can be done in a single sweep.

The cycle \( c \) will be contracted into a super-vertex \( v_c^{(1)} \), which resembles a super-state. The asymptotic dynamics of the resulting coarse-grained Markov jump process will be represented by a smaller graph \( G^{(1)} \). \( G^{(1)} \) will inherit most of the vertices and arcs from \( G \), except that (1) it replaces the subset of vertices in the cycle \( c \) with the super-vertex \( v_c^{(1)} \); (2) all the arcs with both endpoints in \( c \) are discarded; and (3) arcs with tails in \( c \) will have to change their weights. The super-vertex \( v_c^{(1)} \) of \( G^{(1)} \) is allowed to emit multiple arcs with the same head. The logic behind contraction is that if we find the optimal W-graphs \( g_k^{*^{(1)}} \) of \( G^{(1)} \) with sinks fewer than \( k_1 \), we can then expand the super-vertex \( v_c^{(1)} \) back to the cycle \( c \) and cut off a certain arc in the cycle to obtain \( g_k^* \). In essence, such an algorithm is recursive.
2.5.2 The Recursive Structure of \texttt{OptWgraphs()}

The pseudocode for constructing the hierarchy of optimal W-graphs is presented in Algorithm 1, whose main body is the recursive function \texttt{OptWgraphs()}. A flowchart that helps visualize Algorithm 1 is presented in Figure 2.4. To focus on asymptotic analysis, Algorithm 1 presents only the procedures involving exponential factors. The computation of pre-factors $\alpha_k$ can be easily embedded into the pseudocode if needed.

Let $r$ be the depth of recursion and $G^{(r)}$ be the corresponding underlying graph. Assumption 2.5.1 below assures that there is always a unique minimizer in line 8 of Algorithm 1.

\textbf{Assumption 2.5.1 (No symmetry).} \textit{Assume the input graph $G$ is such that, in all depth of recursion of Algorithm 1, every vertex of $G^{(r)}$ has a unique min-outgoing arc and no two min-outgoing arcs are of the same weight.}

Assumption 2.5.1 is stronger than Assumption 2.2.2. No symmetry ensures all optimal W-graphs exist uniquely, but uniqueness of optimal W-graphs can not guarantee no symmetry. Figure 2.3 demonstrates a simple example.

In each recursion of Algorithm 1, the two global variable $\Delta$ and $\bar{k}$ serve as thresholds. The set of min-outgoing arcs are divided into two subsets, those with weights no greater than $\Delta$ constitute the optimal W-graph with $\bar{k}$ sinks, and the rest are candidate arcs for building optimal W-graphs with fewer sinks. $\Delta$ and $\bar{k}$ need to be updated each time before the program enters another recursion. The optimal W-graphs obtained from the while-loop consist of only min-outgoing arcs of
Figure 2.3: An example illustrating that unique optimal W-graphs does not imply no symmetry. Left: an underlying graph $G$ with symmetry, there are two min-outgoing arcs from the vertex 2. Middle and right: the unique optimal W-graphs $g_2^*$ and $g_1^*$. Blue and yellow vertices correspond to sinks and non-sinks, respectively.

the respective underlying graphs. After exiting the while-loop, if no cycle is formed, then the program has arrived at the bottom of recursion and will start backtracking to the top. Otherwise, it contracts the cycle found in $B$ and enters one more level of recursion. We point out that the increments $\Delta_k$ are indifferent to the depth of recursion, meaning, $\Delta_k = \mathcal{V}(g^{*(r)}) - \mathcal{V}(g^{*(r+1)})$ for all $r$.

2.5.3 The Functions Contract_W() and Expand_W()

The pseudocodes for the functions Contract_W() and Expand_W() are presented in page 25. The function Expand_W() is essentially an implementation of (b’) in Section 2.5.1. Inside the function Contract_W(), vertex $i$ of $G$ is mapped to the vertex $i^{(1)}$ of $G^{(1)}$, and $i^{(1)} = i$ if and only if vertex $i$ is not in the cycle $c$, otherwise $i^{(1)} = v_c^{(1)}$ (the super-vertex). In the following, we explain the reason behind the arc-weight-updating procedure of the function Contract_W().

In Algorithm 1, all arcs in the cycle $c$ are min-outgoing arcs of $G$. Denote the
Algorithm 1 Optimal W-graphs

Initialization (global variables): $r = 0$, $\Delta = 0$, $\bar{k} = n$;

Input: $G(S, A, U)$;

Output: $\{g_k^*\}_{k_0-1}^1$, $\{\Delta_k\}_{1}^{k_0-1}$;

1. function OptWgraphs($G$)
2. Prepare the set of min-outgoing arcs $A_0$;
3. $B = \{(i \rightarrow j) \in A_0 \mid U_{ij} \leq \Delta\}$; $B_0 = A_0 \setminus B$;
4. $k_0 = \bar{k}$;
5. $k = k_0$;
6. while no cycle found in $B$ & $|B_0| > 1$ do
7. $k = k - 1$;
8. $(p_k \rightarrow q_k) = \arg \min_{(i \rightarrow j) \in B_0} U_{ij}$;
9. $\Delta_k = U_{p_kq_k}$;
10. $B_0 = B_0 \setminus \{(p_k \rightarrow q_k)\}$;
11. $B = B \cup \{(p_k \rightarrow q_k)\}$;
12. $g_k^* = B$;
13. end while
14. if no cycle found in $B$ then
15. return $\{g_k^*\}_{1}^{k_0-1}$, $\{\Delta_k\}_{1}^{k_0-1}$;
16. else
17. $r = r + 1$;
18. $\bar{k} = k + 1$;
19. $\Delta = \Delta_k$;
20. Find the cycle $c$ formed in $B$;
21. Call $G^{(1)} = \text{Contract}_W(G, c)$;
22. Call $\left(\{g_k^{(1)}\}_{1}^{k_1-1}, \{\Delta_k\}_{1}^{k_1-1}\right) = \text{OptWgraphs}(G^{(1)})$ \hspace{1cm} $\triangleright k_1 = \bar{k}$.
23. for $1 \leq k \leq k_1 - 1$ do
24. Call $g_k^* = \text{Expand}_W(g_k^{(1)}, c)$;
25. end for
26. return $\{g_k^*\}_{1}^{k_0-1}$, $\{\Delta_k\}_{1}^{k_0-1}$;
27. end if
28. end function
Figure 2.4: The flowchart of Algorithm 1. The underlying graphs of each recursion are in the top row. The final depth of recursion is $\hat{r}$. Green boxes contain the optimal W-graphs obtained from the while-loop. Red boxes signal the detection of cycles after exiting the while-loop, and trigger the program to contract cycles and enter one more level of recursion. Yellow boxes contain the optimal W-graphs obtained through expansions.
Input: \( G(S, A, U), c \);
Output: \( G^{(1)}(S^{(1)}, A^{(1)}, U^{(1)}) \);

1: function Contract \( W \) (\( G, c \) )
2: \( S^{(1)} = S \cup \{ v_c^{(1)} \} \);
3: for \( i \in S \) do
4: \( i^{(1)} = i \);
5: \( \gamma(i) = 0 \);
6: if \( i \in c \) then
7: \( S^{(1)} = S^{(1)} \setminus \{ i \} \);
8: \( i^{(1)} = v_c^{(1)} \);
9: \( \gamma(i) = U_{yJ(y)} - U_{iJ(i)} \);
10: end if
11: end for
12: \( A^{(1)} = \emptyset; U^{(1)} = \emptyset \);
13: for \( (i \rightarrow j) \in A \) do
14: if \( i^{(1)} \neq j^{(1)} \) then \( \triangleright \text{ Otherwise, } i^{(1)} = j^{(1)} = v_c^{(1)} \).
15: \( A^{(1)} = A^{(1)} \cup \{ (i^{(1)} \rightarrow j^{(1)}) \} \);
16: \( U_{i^{(1)}j^{(1)}}^{(1)} = \gamma(i) + U_{ij} \);
17: \( U^{(1)} = U^{(1)} \cup \{ U_{i^{(1)}j^{(1)}}^{(1)} \} \);
18: end if
19: end for
20: return \( G^{(1)}(S^{(1)}, A^{(1)}, U^{(1)}) \);
21: end function

---

Input: \( g_k^{(1)}, c \);
Output: \( g_k^* \);

1: function Expand \( W \) (\( g_k^{(1)}, c \) )
2: \( g_k^* = \emptyset \);
3: \( p = y \);
4: for \( (i^{(1)} \rightarrow j^{(1)}) \in g_k^{*(1)} \) do
5: \( g_k^* = g_k^* \cup \{ (i \rightarrow j) \} \);
6: if \( i^{(1)} == v_c^{(1)} \) then
7: \( p = i \);
8: end if
9: end for
10: \( g_k^* = g_k^* \cup \{ (i \rightarrow j) \in c \mid i \neq p \} \);
11: return \( g_k^* \);
12: end function
min-outgoing arc from \(i\) by \((i \rightarrow J(i))\) and its weight by \(U_{iJ(i)}\). Let \(y\) be the tail of the maximum weight arc in the cycle \(c\), i.e., \(y = \arg \max_{i \in c} U_{iJ(i)}\).

Cyclic behavior of a process before exit can be captured asymptotically by the cycle \(c\). Consider the restricted dynamics of rotations along the cycle \(c\) with a \(|c| \times |c|\) generator matrix \(L^c\) defined as follows:

\[
L^c_{ij} = \begin{cases} 
L_{iJ(i)}, & \text{if } j = J(i), \\
-L_{iJ(i)}, & \text{if } j = i, \\
0, & \text{otherwise}. 
\end{cases}
\] (2.8)

The associated invariant distribution \(\pi^c\) is

\[
\pi^c(i) = \frac{L^{-1}_{iJ(i)}}{\sum_{i \in c} L^{-1}_{iJ(i)}} \approx \exp \left( -\gamma(i)/\varepsilon \right), \text{ where } \gamma(i) = U_{yJ(y)} - U_{iJ(i)}, \ i \in c. \tag{2.9}
\]

When a process is rotating along the cycle \(c\), the exit via \((p \rightarrow q)\), where \(p \in c\) and \(q \notin c\), has an effective rate of

\[
\pi^c(p)L_{pq} \approx \exp \left( -\gamma(p) + U_{pq} \right)/\varepsilon). \tag{2.10}
\]

Once we contract the cycle \(c\) into the super-vertex \(v_{(1)c}\), the arc \((p \rightarrow q)\) in \(G\) will resemble itself as an arc in \(G^{(1)}\) with tail \(v_{(1)c}\) and head \(q\). Moreover, it should have an updated weight of \(\gamma(p) + U_{pq}\). On the other hand, for any arc in \(G\) which is not an exit arc from the cycle \(c\), we should not change its weight.

**Theorem 2.5.2.** Under Assumption 2.5.1, \texttt{OptWgraphs()} correctly finds the hierarchy of optimal W-graphs \(\{g^*_{k}\}_{1}^{n-1}\) and the sequence of increments of total weights \(\{\Delta_k\}_{1}^{n-1}\).

The proof of Theorem 2.5.2 is provided in Appendix C.
2.5.4 An Illustrative Example

Figure 2.5 presents a simple example illustrating the key steps of Algorithm 1 in a format that is compatible with the flowchart (Figure 2.4). The input graph $G$ has four vertices, labeled 1, 2, 3, 4, and nine arcs with weights written near by. The final depth of recursion is $\hat{r} = 2$. No symmetry presents at every recursion level.

First, the optimal W-graphs $g^*_3$ of $G$ is obtained without recursion (level zero). It consists of the smallest min-outgoing arc $(1 \rightarrow 2)$ of $G$, and $\Delta_3 = U_{12} = 1$. When the second smallest min-outgoing arc $(2 \rightarrow 1)$ of $G$ is added to $g^*_3$, we do not obtain the optimal W-graph $g^*_2$, instead, the cycle $\{1, 2\}$ is formed. The graph $G$ then contracts the cycle $\{1, 2\}$ into the super-vertex $\{1, 2\}$, resulting in the graph $G^{(1)}$. Arcs coming out from the super-vertex $\{1, 2\}$ of $G^{(1)}$ are originally arcs exiting the cycle $\{1, 2\}$ in $G$, and during the contraction procedure, those of them exiting via the vertex 1 need to increase their weights by $2 - 1 = 1$. Specifically, the arcs $(1 \rightarrow 3)$ and $(1 \rightarrow 4)$ of $G$, represented as $(\{1, 2\} \rightarrow 3)$ and $(\{1, 2\} \rightarrow 4)$ in $G^{(1)}$, have weights $2 + 1 = 3$ and $3 + 1 = 4$ in $G^{(1)}$, respectively. All other arcs in $G^{(1)}$ are inherited from $G$ without modifying weights.

We then enter recursion level one, working on the graph $G^{(1)}$ with three vertices and seven arcs. The optimal W-graph $g^{*(1)}_2$ is obtained, which consists of the smallest min-outgoing arc $(\{1, 2\} \rightarrow 3)$ of $G^{(1)}$, and $\Delta_2 = U^{(1)}_{(1,2)\rightarrow 3} = 3$. When the second smallest min-outgoing arc $(3 \rightarrow \{1, 2\})$ is added to $g^{*(1)}_2$, we do not obtain the optimal W-graph $g^{*(1)}_1$, instead, the cycle $\{1, 2, 3\}$ is formed. The graph $G^{(1)}$ then contracts the cycle $\{1, 2, 3\}$ into the super-vertex $\{1, 2, 3\}$, resulting in the graph...
$G^{(2)}$. Arcs coming out from the super-vertex $\{1, 2, 3\}$ in $G^{(2)}$ are originally arcs exiting the cycle $\{1, 2, 3\}$ in $G^{(1)}$, and during the contraction procedure, those of them exiting via the super-vertex $\{1, 2\}$ need to increase their weights by $4 - 3 = 1$. Specifically, the arc $(\{1, 2\} \to 4)$ of $G^{(1)}$, represented as $(\{1, 2, 3\} \to 4)$ in $G^{(2)}$, has weight $4 + 1 = 5$ in $G^{(2)}$. All other arcs in $G^{(2)}$ are inherited from $G^{(1)}$ without modifying weights.

We then enter recursion level two, working on the graph $G^{(2)}$ with two vertices and three arcs. The optimal $W$-graph $g_1^*(2)$ is obtained, which consists of the smallest min-outgoing arc $(\{1, 2, 3\} \to 4)$ of $G^{(2)}$, and $\Delta_1 = U_{\{1,2,3\} \to 4}^{(2)} = 5$. Here we have arrived at the bottom of recursion and will start backtracking to the top.

During the backtracking process, super-vertices are expanded back to cycles and one arc in each cycle needs to be cut off. First, the optimal $W$-graph $g_1^*(2)$ is expanded to the optimal $W$-graph $g_1^*(1)$ at recursion level one, where the arc $(\{1, 2, 3\} \to 4)$ is mapped back to the arc $(\{1, 2\} \to 4)$, and the arc $(\{1, 2\} \to 3)$ in the cycle $\{\{1, 2\}, 3\}$ is cut off. Next, at recursion level zero, the optimal $W$-graphs $g_2^*(1)$ and $g_1^*(1)$ are expanded to the optimal $W$-graphs $g_2^*$ and $g_1^*$, respectively. From $g_2^*(1)$ to $g_2^*$, the arc $(\{1, 2\} \to 3)$ is mapped back to the arc $(1 \to 3)$, and the arc $(1 \to 2)$ in the cycle $\{1, 2\}$ is cut off. From $g_1^*(1)$ to $g_1^*$, the arcs $(3 \to \{1, 2\})$ and $(\{1, 2\} \to 4)$ are mapped back to the arcs $(3 \to 2)$ and $(1 \to 4)$, respectively, and the arc $(1 \to 2)$ in the cycle $\{1, 2\}$ is cut off.
Figure 2.5: An illustrative example of Algorithm 1. Green arcs are min-outgoing arcs from each vertex/super-vertex. Red arcs form the cycles encountered in each recursion. Arc weights colored in red indicate strictly increasing after contractions. Blue and yellow vertices correspond to sinks and non-sinks respectively.
Chapter 3: T-graphs and Asymptotic Eigenvectors

Assumption 2.5.1 (no symmetry) in Section 2.5.2 is rather restrictive. Many models arising from applications often present symmetry. In fact, Algorithm 1 performs properly for any input graph $G$ that is strongly connected if we do the following: (1) in line 2 of Algorithm 1, when there are multiple min-outgoing arcs from the same vertex, select one arbitrarily to put into $A_0$; and (2) in line 8 of Algorithm 1, when there are more than one minimizer in $B_0$, pick one arbitrarily to be $(p_k \rightarrow q_k)$. In general, Algorithm 1 produces a hierarchy of optimal W-graphs which satisfies the weak nested property. If optimal W-graphs are not unique, different runs of Algorithm 1 could produce different hierarchies of optimal W-graphs, however, the collections of increments $\{\Delta_k\}_{k=1}^{n-1}$ are always the same.

Nonetheless, optimal W-graphs do not capture possible cyclic behavior before exits, nor do they inform us about the effective exit rates. In this chapter, we introduce the concepts of the hierarchy of Typical Transition Graphs (T-graphs) and the associated sequence of Characteristic Timescales. A T-graphs $g(\Gamma_h)$ records all typical transitions (in the logarithmic asymptotic sense) as well as their effective transition rates up to the characteristic timescale $t(\Gamma_h) \simeq \exp(\Gamma_h/\varepsilon)$.

In section 3.1, the hierarchy of T-graphs and the associated characteristic
timescales are defined inductively via Algorithm 2, which is indeed a modification of Algorithm 1. It is worth noting that Algorithm 2 makes no assumption on its input graph. In other words, the hierarchy of T-graphs is well defined for any weighted directed graph \( G \).

Although the constructions of T-graphs share many of the key steps with the constructions of optimal W-graphs, T-graphs differ from optimal W-graphs by two distinctive features. First, T-graphs contain communicating classes. Secondly, vertices in T-graphs also carry weights. In section 3.2, we identify the closed communicating classes in T-graphs with Freidlin’s hierarchy of Markov chains. In section 3.3, we discuss why the closed communicating classes in each T-graph can be viewed as metastable classes between two characteristic timescales, and how the weights on vertices can be used to estimate metastable distributions on these closed communicating classes.

Under the assumption that all optimal W-graphs are unique, we construct asymptotic estimates of eigenvectors using T-graphs in section 3.4, where the weak nested property is further exploited. Metastable distributions on the collapsing and absorbing closed communicating classes are used to construct asymptotic estimates of left eigenvectors, while the asymptotic estimates of right eigenvectors are expressed as indicator functions of the collapsing connected components.

In section 3.5, we describe an object-oriented, non-recursive implementation scheme for constructing the hierarchy of T-graphs.
3.1 The Hierarchy of Typical Transition Graphs and Characteristic Timescales

A T-graph \( g(\Gamma_h) \) is a weighted directed graph with weights on both arcs and vertices. When vertex weights are ignored, it is a spanning subgraph of \( G(S,A,U) \), i.e., it has vertex set \( S_h = S \), arc set \( A_h \subset A \) and arc weights \( U_{ij} \). Each vertex \( i \) in \( g(\Gamma_h) \) also carries a weight \( \delta_h(i) \). An arc \( (i \to j) \) in \( A_h \) has an effective weight of \( \delta_h(i) + U_{ij} \), indicating an effective transition rate of the order of \( \exp(- (\delta_h(i) + U_{ij}) / \varepsilon) \).

We will identify a T-graph \( g(\Gamma_h) \) with its arc set \( A_h \) and vertex-weight vector \( \delta_h \) by denoting \( g(\Gamma_h) = (A_h, \delta_h) \). Index \( h = 0, 1, \ldots, \hat{h} \), where the total number of T-graphs \( \hat{h} \) is not known ahead. By construction, (1) \( \Gamma_0 = 0 \) and the T-graph \( g(\Gamma_0) = (\emptyset, 0) \); and (2) \( \Gamma_h < \Gamma_{h+1} \), \( A_h \subset A_{h+1} \), and \( \delta_h \leq \delta_{h+1} \) (entry-wise). In this sense, the hierarchy of T-graphs are strongly nested.

3.1.1 The Recursive Structure of Tgraphs()

The pseudocode for constructing the hierarchy of T-graphs is presented in Algorithm 2, whose main body is the recursive function \texttt{Tgraphs()}\. A flowchart that helps visualize Algorithm 2 is presented in Figure 3.1.

In Algorithm 2, \( A_0 \) is the subset of \( A \) consisting of all min-outgoing arcs from each vertex, including multiples. Each while-loop of Algorithm 2 processes a batch of min-arcs \( B_{\text{min}} \), while each while-loop of Algorithm 1 processes a single min-arc. Adding multiple arcs to \( B \) simultaneously may create multiple closed communicating
classes, accordingly, we trace them all in Algorithm 2. \( \gamma \) in Algorithm 2 is an auxiliary variable for calculating the vertex-weight vector \( \delta_h \). It is the same as the \( \gamma \) inside the function \textbf{Contract}_W().

Similar to Algorithm 1, in each recursion of Algorithm 2, the two global variables \( \Gamma \) and \( h \) serve as thresholds. The set of min-outgoing arcs are divided into two subsets, those with weights no greater than \( \Gamma \) constitute the \( T \)-graph corresponding to the \( h \)th characteristic timescale, and the rest are candidates arcs for building the \( T \)-graphs corresponding to subsequent characteristic timescales. \( \Gamma \) and \( h \) need to be updated each time before the program enters another recursion. The \( T \)-graphs obtained from the while-loop consist of only min-outgoing arcs of the respective underlying graphs. After exiting the while-loop, if all vertices belong to a single communicating class, then the program has arrived at the bottom of recursion and will start backtracking to the top. Otherwise, it contracts all the closed communicating classes found in \( B \) and enters one more level of recursion. We point out that the exponential factors \( \Gamma_h \) of the characteristic timescales are indifferent to the depth of recursion.

3.1.2 The Functions \textbf{Contract}_T() and \textbf{Expand}_T()

The pseudocodes for the functions \textbf{Contract}_T() and \textbf{Expand}_T() are presented in pages 37 and 38. The function \textbf{Contract}_T() essentially follows the same principle as the function \textbf{Contract}_W(), however, instead of contracting a single cycle, it contracts multiple closed communicating classes. Vertex \( i \) of \( G \) is mapped to the vertex
Algorithm 2 T-graphs

Initialization (global variables): \( r = 0, \Gamma = 0, h = 0; \)

Input: \( G(S, A, U); \)

Output: \( \{g(\Gamma_h)\}_{h_0}^{\hat{h}}, \{\Gamma_h\}_{h_0}^{\hat{h}}; \)

1: function \( \text{Tgraphs} \left( G \right) \)
2: \( \) Prepare the set of min-outgoing arcs \( A_0; \)
3: \( B = \{(i \rightarrow j) \in A_0 \mid U_{ij} \leq \Gamma\}; \)
4: \( B_0 = A_0 \setminus B; \)
5: \( h_0 = h; \)
6: \( h = h_0; \)
7: \( \) \textbf{while} no closed communicating classes found in \( B \) \& \( |B_0| > 0 \) \textbf{do}
8: \hfill \( h = h + 1; \)
9: \hfill \( \Gamma_h = \min_{(i \rightarrow j) \in B_0} U_{ij}; \)
10: \hfill \( B_{\text{min}} = \{(i \rightarrow j) \in B_0 \mid U_{ij} = \Gamma_h\}; \)
11: \hfill \( B_0 = B_0 \setminus B_{\text{min}}; \)
12: \hfill \( B = B \cup B_{\text{min}}; \)
13: \hfill \( A_h = B; \)
14: \hfill \( \delta_h = 0; \)
15: \hfill \( g(\Gamma_h) = (A_h, \delta_h); \)
16: \textbf{end while}
17: \hfill \( r = r + 1; \)
18: \hfill \( h = h; \)
19: \hfill \( \Gamma = \Gamma_h; \)
20: \hfill \) Find all closed communicating classes \( \{c_d\}_{1}^{d_0} \) formed in \( B; \)
21: \hfill \) Call \( \left( G^{(1)}, \gamma \right) = \text{Contract}_{T}( G, \{c_d\}_{1}^{d_0}); \)
22: \hfill \( \delta_h = \gamma; \)
23: \hfill \) Call \( \left( \{g^{(1)}(\Gamma_h)\}_{h_1}^{h_1+1}, \{\Gamma_h\}_{h_1}^{h_1+1} \right) = \text{Tgraphs}( G^{(1)}); \)
24: \hfill \( h_1 = \hat{h}. \)
25: \hfill \) \textbf{for} \( h_1 + 1 \leq h \leq \hat{h} \) \textbf{do}
26: \hfill \) Call \( g(\Gamma_h) = \text{Expand}_{T}( g^{(1)}(\Gamma_h), \{c_d\}_{1}^{d_0}, \gamma); \)
27: \textbf{end for}
28: \hfill \) return \( \{g(\Gamma_h)\}_{h_0}^{\hat{h}}, \{\Gamma_h\}_{h_0}^{\hat{h}}; \)
29: \hfill \) \textbf{end if}
30: \textbf{end function}
Figure 3.1: The flowchart of Algorithm 2. The underlying graphs of each recursion are in the top row. The final depth of recursion is $\hat{r}$. Green and blue boxes contain the T-graphs obtained from the while-loop. In particular, blue boxes signal the detection of closed communicating classes in the last T-graph obtained from the while-loop and the condition for if is not satisfied, which then trigger the program to contract all closed communicating classes encountered and enter another recursion. Yellow boxes contain the T-graphs obtained through expansions.
\(i^{(1)}\) of \(G^{(1)}\), and \(i^{(1)} = i\) if and only if vertex \(i\) is not in any closed communicating class \(c_d\) encountered by Algorithm 2, otherwise \(i^{(1)} = v_{i_d}^{(1)}\) (a super-vertex) for some \(d\).

The critical difference between these two contraction functions is that the function \(\text{Contract}_T()\) also returns the auxiliary variable \(\gamma\) which is used to update the arc weights. In fact, the arc-weight-updating procedure specifies a recurrence relationship on arc weights: for any arc \((i^{(r)} \rightarrow j^{(r)})\) in \(G^{(r)}\),

\[
U_{i^{(r)}j^{(r)}}^{(r)} = \gamma^{(r-1)}(i^{(r-1)}) + U_{i^{(r-1)}j^{(r-1)}}^{(r-1)} = \cdots = \sum_{r'=0}^{r-1} \gamma^{(r')}(i^{(r')}) + U_{ij}. \tag{3.1}
\]

Different from the function \(\text{Expand}_W()\), the function \(\text{Expand}_T()\) preserves all the arcs in the closed communicating classes so as to capture cyclic behavior. Moreover, it requires the crucial input of the auxiliary variable \(\gamma\) to compute the vertex-weight vector \(\delta_h\), and such vertex-weight-updating procedure specifies a recurrence relationship on vertex weights: for any vertex \(i\) in the T-graphs \(g(\Gamma_h)\),

\[
\delta_h(i) = \delta_h^{(1)}(i^{(1)}) + \gamma(i) = \cdots = \delta_h^{(r)}(i^{(r)}) + \sum_{r'=0}^{r-1} \gamma^{(r')}(i^{(r')}). \tag{3.2}
\]

From the construction of Algorithm 2, we know that for any \(h\), a recursion level \(r_h\) can be found such that the T-graph \(g^{(r_h)}(\Gamma_h)\) contains no closed communicating classes. Consequently, the vertex-weight vector \(\delta_h^{(r_h)} = 0\), thus any arc \((i^{(r_h)} \rightarrow j^{(r_h)})\) in the T-graph \(g^{(r_h)}(\Gamma_h)\) has an effective weight of

\[
\delta_h^{(r_h)}(i^{(r_h)}) + U_{i^{(r_h)}j^{(r_h)}}^{(r_h)} = U_{i^{(r_h)}j^{(r_h)}}^{(r_h)}. \tag{3.3}
\]

On the other hand, an arc \((i^{(r_h)} \rightarrow j^{(r_h)})\) of the T-graph \(g^{(r_h)}(\Gamma_h)\) will be mapped recursively back to the arc \((i \rightarrow j)\) of the T-graph \(g(\Gamma_h)\), which has an
effective weight of

\[ \delta_h(i) + U_{ij} \overset{\text{Eq.\ 3.2}}{=} \sum_{r'=0}^{r_h-1} \gamma(r')(i(r')) + U_{ij} \overset{\text{Eq.\ 3.1}}{=} U_{i(r_h)j(r_h)}. \] (3.4)

Input: \( G(S, A, U) \), \( \{c_d\}_{d_0} \)
Output: \( G^{(1)}(S^{(1)}, A^{(1)}, U^{(1)}), \gamma \)

1: function \( \text{Contract}_T(G, \{c_d\}_{d_0}) \)
2: \( S^{(1)} = S \cup \{v^{(1)}_{cd}\}_{1} \)
3: for \( i \in S \) do
4: \( i^{(1)} = i \)
5: \( \gamma(i) = 0 \)
6: for \( 1 \leq d \leq d_0 \) do
7: if \( i \in c_d \) then
8: \( S^{(1)} = S^{(1)} \setminus \{i\} \)
9: \( i^{(1)} = v^{(1)}_{cd} \)
10: \( \gamma(i) = U_{y_dJ(y_d)} - U_{iJ(i)} \)
11: break
12: end if
13: end for
14: end for
15: \( A^{(1)} = \emptyset \); \( U^{(1)} = \emptyset \)
16: for \( (i \rightarrow j) \in A \) do
17: if \( i^{(1)} \neq j^{(1)} \) then \( \triangleright \) Otherwise, \( i^{(1)} = j^{(1)} = v^{(1)}_{cd} \) for some \( d \)
18: \( A^{(1)} = A^{(1)} \cup \{(i^{(1)} \rightarrow j^{(1)})\} \)
19: \( U^{(1)}_{i^{(1)}j^{(1)}} = \gamma(i) + U_{ij} \) \( \triangleright \) Update weights.
20: \( U^{(1)} = U^{(1)} \cup \{U^{(1)}_{i^{(1)}j^{(1)}}\} \)
21: end if
22: end for
23: return \( G^{(1)}(S^{(1)}, A^{(1)}, U^{(1)}), \gamma \) \( \triangleright \) Return also \( \gamma \).
24: end function

3.1.3 An Illustrative example

Figure 3.2 presents a simple example illustrating the key steps of Algorithm 2 in a format that is compatible with the flowchart (Figure 3.1). The input graph \( G \) has four vertices, labeled 1, 2, 3, 4, and nine arcs with weights written near by. The final depth of recursion is \( \hat{r} = 2 \). Symmetry presents at both recursion levels.
Input: \( g^{(1)}(\Gamma_h) \), \( \{c_d\}^d_1 \), \( \gamma \);  
Output: \( g(\Gamma_h) \);  

1: **function** Expand_T( \( g^{(1)}(\Gamma_h) \), \( \{c_d\}^d_1 \), \( \gamma \) )  
2: \( A_h = \emptyset \);  
3: **for** \( (i^{(1)} \rightarrow j^{(1)}) \in g^{(1)}(\Gamma_h) \) **do**  
4: \( A_h = A_h \cup \{(i \rightarrow j)\} \);  
5: **if** \( i^{(1)} = i \) **then**  
6: \( \delta_h(i) = \delta^{(1)}_h(i) \);  
7: **end if**  
8: **end for**  
9: **for** \( 1 \leq d \leq d_0 \) **do**  
10: \( A_h = A_h \cup \{(i \rightarrow j) \in c_d\} \);  
11: **for** \( i \in c_d \)  
12: \( \delta_h(i) = \delta^{(1)}_h(i^{(1)}) + \gamma(i) \);  
13: **end for**  
14: **end for**  
15: \( g(\Gamma_h) = (A_h, \delta_h) \);  
16: **return** \( g(\Gamma_h) \);  
17: **end function**

First, the T-graph \( g(\Gamma_1) \) is obtained without recursion (level zero). It consists of the two smallest min-outgoing arcs \((1 \rightarrow 2)\) and \((2 \rightarrow 1)\), and \( \Gamma_1 = U_{12} = U_{21} = 1 \). These two arcs form the closed communicating class \( \{1, 2\} \). The graph \( G \) then contracts the communicating class \( \{1, 2\} \) into the super-vertex \( \{1, 2\} \), resulting in the graph \( G^{(1)} \). Arcs coming out from the super-vertex \( \{1, 2\} \) in \( G^{(1)} \) are originally arcs exiting the communicating class \( \{1, 2\} \) in \( G \). Since all arcs in the communicating class \( \{1, 2\} \) are of equal weights, no arcs in \( G^{(1)} \) increase their weights, and all vertices in \( g(\Gamma_1) \) have weights 0.

We then enter recursion level one, working on the graph \( G^{(1)} \) with three vertices and seven arcs. First, the T-graph \( g^{(1)}(\Gamma_2) \) is obtained, which consists of the first min-outgoing arc \((\{1, 2\} \rightarrow 3)\) of \( G^{(1)} \), and \( \Gamma_2 = U_{\{1, 2\} \rightarrow 3} = 2 \). Since no closed communicating classes are formed, all vertices in \( g^{(1)}(\Gamma_2) \) have weights 0.
When the two second smallest min-outgoing arcs (3 → \{1, 2\}) (multiples) of \(G^{(1)}\) are added to the T-graph \(g^{(1)}(\Gamma_2)\), we obtain the T-graph \(g^{(1)}(\Gamma_3)\), and \(\Gamma_3 = U_{3\rightarrow\{1,2\}}^{(1)} = 5\). The T-graph \(g^{(1)}(\Gamma_3)\) contains a closed communicating class \{"1, 2\}, 3\}. The graph \(G^{(1)}\) then contracts this communicating class into the super-vertex \{1, 2, 3\}, resulting in the graph \(G^{(2)}\). Arcs coming out from the super-vertex \{1, 2, 3\} in \(G^{(2)}\) are originally arcs exiting the communicating class \{"1, 2\}, 3\} in \(G^{(1)}\). Since arcs in the communicating class \{"1, 2\}, 3\} are of different weights, arcs exiting via the super-vertex \{1, 2\} need to increase their weights by \(5 - 2 = 3\). Specifically, the arc (\{1, 2\} → 4) of \(G^{(1)}\), represented as (\{1, 2, 3\} → 4) in \(G^{(2)}\), has weight \(3 + 3 = 6\) in \(G^{(2)}\). All other arcs in \(G^{(2)}\) are inherited from \(G^{(1)}\) without modifying weights. Meanwhile, in the T-graph \(g^{(1)}(\Gamma_3)\), the super-vertex \{1, 2\} needs to increase its weight to \(0 + 5 - 2 = 3\), and other vertices keep their weights unchanged.

We then enter recursion level two, working on the graph \(G^{(2)}\) with two vertices and three arcs. The T-graph \(g^{(2)}(\Gamma_4)\) is obtained, which consists of the two smallest min-outgoing arcs (\{1, 2, 3\} → 4) and (4 → \{1, 2, 3\}) of \(G^{(2)}\), and \(\Gamma_4 = U_{\{1,2,3\}\rightarrow4}^{(2)} = U_{4\rightarrow\{1,2,3\}}^{(2)} = 6\). The T-graph \(g^{(2)}(\Gamma_4)\) is a single communicating class \{"1, 2, 3\}, 4\}, so we have arrived at the bottom of recursion and will start backtracking to the top. Since all arcs in the communicating class \{"1, 2, 3\}, 4\} are of equal weights, all vertices in \(g^{(2)}(\Gamma_4)\) have weights 0.

During the backtracking process, super-vertices are expanded back to communicating classes. At the same time, each vertex in the respective communicating classes will be assigned a weight. First, the T-graph \(g^{(2)}(\Gamma_4)\) is expanded to
the T-graph $g^{(1)}(\Gamma_4)$ at recursion level one, where the arcs \(\{1, 2, 3\} \rightarrow 4\) and \(4 \rightarrow \{1, 2, 3\}\) are mapped back to the arcs \(\{1, 2\} \rightarrow 4\) and \(4 \rightarrow 3\), respectively, and the full communicating class \(\{1, 2\}, 3\) is recovered. Vertex 3 is assigned a weight of \(0+5−5 = 0\) and the super-vertex \(\{1, 2\}\) is assigned a weight of \(0+5−2 = 3\).

Next, at recursion level zero, the T-graphs $g^{(1)}(\Gamma_2)$, $g^{(1)}(\Gamma_3)$, and $g^{(1)}(\Gamma_4)$ are expanded to the T-graphs $g(\Gamma_2)$, $g(\Gamma_3)$, and $g(\Gamma_4)$, respectively. All expansions recover the full communicating class \(\{1, 2\}\). The arc \(\{1, 2\} \rightarrow 3\) of $G^{(1)}$ is mapped back to the arc \(1 \rightarrow 3\) of $G$, the two arcs \(3 \rightarrow \{1, 2\}\) of $G^{(1)}$ are mapped back to the arcs \(3 \rightarrow 2\) and \(3 \rightarrow 1\) of $G$, respectively, and the arc \(\{1, 2\} \rightarrow 4\) of $G^{(1)}$ is mapped back to \(1 \rightarrow 4\) of $G$. In the T-graph $g(\Gamma_2)$, both vertices 1 and 2 are assigned a weight of \(0 + 1 − 1 = 0\). In the T-graphs $g(\Gamma_3)$ and $g(\Gamma_4)$, both vertices 1 and 2 are assigned a weight of \(3 + 1 − 1 = 3\).

3.2 Communicating Classes in T-graphs and Freidlin’s Hierarchy of Markov Chains

In this section, we discuss the relationship between the communicating classes in T-graphs and Freidlin’s hierarchy of Markov Chains.

Let \(\{c_{d}^{(r)}\}_{d=1}^{\hat{r}}\), where \(r = 0, \ldots, \hat{r}\), be the collection of closed communicating classes encountered in the \(r\)th recursion of Algorithm 2. For each \(c_{d}^{(r)}\), let \(C_{r,d}\) be its full expansion in $G$, so \(C_{r,d}\) contains no super-vertices. The collection of closed communicating classes \(\{C_{r,d}\}_{d=1}^{\hat{r}}\) share the same birth timescale \(t(\Gamma_{h_{r+1}}) \asymp \exp(\Gamma_{h_{r+1}}/\varepsilon)\), which is also the birth timescale of the T-graph $g(\Gamma_{h_{r+1}})$. 

40
Figure 3.2: An illustrative example of Algorithm 2. Green arcs are min-outgoing arcs from each vertex/super-vertex. Vertex weights are put inside the brackets near by. Arc and vertex weights colored in red indicate strictly increasing after contractions and expansions. Blue and yellow vertices correspond to recurrent and transient states, respectively. The closed communicating classes encountered by Algorithm 2 are \{1, 2\}, \{1, 2, 3\}, and \{1, 2, 3, 4\}. The closed communicating classes in each T-graph are: \{1, 2\}, \{3\}, and \{4\} in \(g(\Gamma_1)\); \{3\} and \{4\} in \(g(\Gamma_2)\); \{1, 2, 3\} and \{4\} in \(g(\Gamma_3)\); and \{1, 2, 3, 4\} in \(g(\Gamma_4)\).
In the illustrative example of Figure 3.2, \( d_0 = 1, \) \( C_{0,1} = \{1, 2\}; \) \( d_1 = 1, \) \( C_{1,1} = \{1, 2, 3\}; \) and \( d_2 = 1, \) \( C_{2,1} = \{1, 2, 3, 4\}. \)

Corollary 3.2.1 below is a direct consequence of the construction procedures of Algorithm 2.

**Corollary 3.2.1.**  
(i) If \( d \neq d' \), then \( C_{r,d} \cap C_{r,d'} = \emptyset. \)

(ii) If \( r < r' \), then either \( C_{r,d} \cap C_{r',d'} = \emptyset \) or \( C_{r,d} \subset C_{r',d'}. \)

In early 1970s, Freidlin proposed to describe the long-term behavior of a metastable process via a hierarchy of cycles [14, 15, 17] in the case of no symmetry. This hierarchy can be mapped to a tree. In the reversible case, this tree is a complete binary tree [6]. Later, in 2014, Freidlin extend this approach to the case with symmetry [16], where the hierarchy of cycles are replaced by the hierarchy of Markov chains. Each Markov chain in Freidlin’s hierarchy is born on a particular timescale, which is the inverse of its rotation rate. [18] introduces the notion of hierarchy of Markov chains in a general setting and calculate the metastable distributions. There, transition rates are not required to be of exponential order (or have any specific asymptotic behavior), rather, the Markov chains are assumed to be *asymptotically regular*, a certain asymptotic relation between the ratios of transition rates.

In our setting of transition rates (Eq. (1.1)), Freidlin’s hierarchy of Markov chains can be constructed as follows. Every vertex \( i \in S \) is a rank-0 Freidlin’s Markov chain (singleton). Let \( g_F \) be the spanning subgraph of \( G(S, A, U) \) with arc set \( \tilde{A}_F = A_0 \), where \( A_0 \) is the subset of \( A \) consisting of all min-outgoing arcs from
each vertex of $G$, including multiples. All the communicating classes in $g_F$ are called rank-1 Freidlin’s Markov chains. If all the vertices of $G$ belong to a single rank-1 Freidlin’s Markov chain, then the process terminates.

Otherwise, let $\{\tilde{c}_l\}_{l=1}^{\tilde{m}_0}$ denote the sub-collection of closed rank-1 Freidlin’s Markov chains. Apply the contraction rules of $\text{Contract}_T()$ to $\{\tilde{c}_l\}_{l=1}^{\tilde{m}_0}$ with $G$ and denote the resulting graph by $G_F^{(1)}(S_F^{(1)},A_F^{(1)},U_F^{(1)})$. Let $g_F^{(1)}$ be the spanning subgraph of $G_F^{(1)}$ with arc set $\tilde{A}_F^{(1)}$, where $\tilde{A}_F^{(1)}$ is the subset of $A_F^{(1)}$ consisting of all min-outgoing arcs from each vertex of $G_F^{(1)}$, including multiples. All the communicating classes in $g_F^{(1)}$ are called rank-2 Freidlin’s Markov chains. If all the vertices of $G_F^{(1)}$ belong to a single rank-2 Freidlin’s Markov chain, then the process terminates. Otherwise, let $\{\tilde{c}_l^{(1)}\}_{l=1}^{\tilde{m}_1}$ denote the sub-collection of closed rank-2 Freidlin’s Markov chains. Again, apply the contraction rules of $\text{Contract}_T()$ to $\{\tilde{c}_l^{(1)}\}_{l=1}^{\tilde{m}_1}$ with $G_F^{(1)}$ and denote the resulting graph by $G_F^{(2)}(S_F^{(2)},A_F^{(2)},U_F^{(2)})$.

Such a recursive procedure will terminate at $G_F^{(\hat{m})}$ for some finite number $\hat{m} \geq 1$. Let $\{\tilde{c}_l^{(m)}\}_{l=1}^{\tilde{m}_m}$, where $m = 0, \cdots, \hat{m}$, denote the sub-collection of closed rank-$(m + 1)$ Freidlin’s Markov chains. For each $\tilde{c}_l^{(m)}$, let $\tilde{C}_{m,l}$ be its full expansion in $G$. Note that the collection of rank-$(m + 1)$ Freidlin’s Markov chains $\{\tilde{C}_{m,l}\}_{l=1}^{\tilde{m}_m}$ may have various birth timescales, in other words, they are not necessarily in sync. Figure 3.3 shows a simple example.

The following Corollary 3.2.2 is a direct consequence of the above construction.

**Corollary 3.2.2.** (i) If $l \neq l'$, then $\tilde{C}_{m,l} \cap \tilde{C}_{m,l'} = \emptyset$.

(ii) If $m < m'$, either $\tilde{C}_{m,l} \cap \tilde{C}_{m',l'} = \emptyset$, or $\tilde{C}_{m,l} \subset \tilde{C}_{m',l'}$. 

43
Figure 3.3: An example of Freidlin’s hierarchy of Markov Chains. The two rank-1 Freidlin’s Markov chains \{1, 2\} and \{3, 4\} have birth timescales of order of \(\exp(1/\varepsilon)\) and \(\exp(2/\varepsilon)\), respectively. Green arcs are min-outgoing arcs from each vertex.

Figure 3.4 illustrates the hierarchy of Freidlin’s Markov chains for the graph \(G\) in Figure 3.2.

Let \(\mathcal{C} = \bigcup_{r=0}^{\hat{r}} \mathcal{C}_r\), where \(\mathcal{C}_r = \{C_{r,d}\}_{d=1}^{d_r}\), be the collection of all closed communicating classes encountered by Algorithm 2, and let \(\mathcal{\tilde{C}} = \bigcup_{m=0}^{\hat{m}} \{\mathcal{\tilde{C}}_{m,l}\}_{l=1}^{l_m}\) be the collection of all closed Freidlin’s Markov chains in each rank. For \(r = 0, \cdots, \hat{r}\), let \(\mathcal{\tilde{C}}_r = \{\mathcal{\tilde{C}}_{m,l} \in \mathcal{\tilde{C}} \mid \text{birth timescale of } \mathcal{\tilde{C}}_{m,l} \asymp \exp(\Gamma h_{r+1}/\varepsilon)\}\) be the \(r\)th synchronized sub-collection of \(\mathcal{\tilde{C}}\).

**Corollary 3.2.3.** (i) \(\mathcal{C}_r = \mathcal{\tilde{C}}_r\); (ii) \(\mathcal{\tilde{C}} = \bigcup_{r=0}^{\hat{r}-1} \mathcal{\tilde{C}}_r\); (iii) \(\mathcal{C} = \mathcal{\tilde{C}}\).

In each communicating class, a subset of states are called main states \cite{14} in the sense that if a process is wondering inside this communicating class, then with probability of order one, it will be found at some main state.

### 3.3 Metastability and T-graphs

The collections of closed communicating classes in T-graphs are central to the understanding of metastability. Let \(\{M_{h,b}\}_{b=1}^{b_h}\) be the collection of closed commu-
Figure 3.4: Freidlin’s hierarchy of Markov Chains for the graph $G$ in Figure 3.2. The closed Freidlin’s Markov chains in each rank are $\{1, 2\}$, $\{1, 2, 3\}$, and $\{1, 2, 3, 4\}$, coincide with the closed communicating classes encountered by Algorithm 2. (See Figure 3.2.) Green arcs are min-outgoing arcs from each vertex/super-vertex. Blue and yellow vertices correspond to recurrent and transient states respectively.
nicating classes in the T-graph \( g(\Gamma_h) \), including singletons. Then \( \{M_{h,b}\}_{b=1}^{b_h} \) can be seen as the collection of metastable classes for the time span \((t(\Gamma_h), t(\Gamma_{h+1}))\). On one hand, with probability tends to one as \( \varepsilon \to 0 \), depending on where it starts, a process will be observed wondering inside one of the \( M_{h,b} \)'s on any timescale \( t \in (t(\Gamma_h), t(\Gamma_{h+1})) \). On the other hand, every metastable class \( M_{h,b} \) has a birth timescale no greater than \( t(\Gamma_h) \) and an exit rate no smaller than \( t^{-1}(\Gamma_{h+1}) \).

On each metastable class \( M_{h,b} \), there exists a metastable distribution \( \mu_{M_{h,b}} \), which is valid from the timescale it is born till the timescale it collapses. The asymptotic estimate of \( \mu_{M_{h,b}} \) can be calculated using the vertex-weight vector \( \delta_h \) by Eq. (3.5),

\[
\mu_{M_{h,b}}(i) \asymp \exp(-\delta_h(i)/\varepsilon), \quad \forall i \in M_{h,b}.
\] (3.5)

The exit rate from the metastable class \( M_{h,b} \) can also be estimated and is given by Eq. (3.6),

\[
e(M_{h,b}) = \sum_{(i \rightarrow j) \in A_{h,b}} \mu_{M_{h,b}}(i)L_{ij} \asymp \exp\{-\min_{(i \rightarrow j) \in A_{h,b}} (\delta_h(i) + U_{ij})/\varepsilon\},
\] (3.6)

where \( A_{h,b} = \{(i \rightarrow j) \in A \mid i \in M_{h,b}, \ j \notin M_{h,b}\} \).

By construction, the number of closed communicating classes in the hierarchy of T-graphs are non-increasing, i.e., \( b_h \geq b_{h+1} \), where \( h = 0, \ldots, \hat{h} - 1 \). In the case where \( b_h > b_{h+1} \), there are \((b_h - b_{h+1})\) closed communicating classes collapsing on the timescale \( t(\Gamma_{h+1}) \asymp \exp(\Gamma_{h+1}/\varepsilon) \). In other words, \((b_h - b_{h+1})\) of the metastable classes \( \{M_{h,b}\}_{b=1}^{b_h} \) have exit rates of the order of \( \exp(-\Gamma_{h+1}/\varepsilon) \). Moreover, for all integers \( k \) such that \( b_h - 1 \geq k \geq b_{h+1} \), increments \( \Delta_k = \Gamma_{h+1} \), so \( \Re(\lambda_k) \asymp \)
exp\((-\Gamma_{h+1}/\varepsilon)\) (Theorem 2.2.1), and all feasible optimal W-graphs $g_k^*$ are covered by the T-graph $g(\Gamma_{h+1})$.

3.4 Asymptotic Estimates of Eigenvectors

Sharp asymptotic estimates of right eigenvectors were obtained in [3] for a certain class of reversible Markov chains. These results are readily applicable to the type of Markov jump processes considered here under the assumption of reversibility and uniqueness of optimal W-graphs. In this case, the $k$th right eigenvector is approximated by the indicator function of the collapsing in-tree $S_k^+$. In this section, we show that such approximation can be extended to the case of irreversible processes. Furthermore, with the knowledge of metastable distributions on closed communicating classes of T-graphs, we are able to obtain asymptotic estimates of left eigenvectors.

When all optimal W-graphs are unique, the increments $\Delta_k$’s are distinct and asymptotic estimates of eigenvalues are given by $\lambda_k \approx \exp(-\Delta_k/\varepsilon)$, $k = 1, \cdots, n-1$ (Theorem 2.2.1 and 2.2.3). Moreover, for each $k$ there exists a unique index $h(k)$ such that the numbers of closed communicating classes in the T-graphs $g(\Gamma_{h(k)-1})$ and $g(\Gamma_{h(k)})$ are $k + 1$ and $k$, respectively, and the increment $\Delta_k = \Gamma_{h(k)}$. The optimal W-graphs $g_{k+1}^*$ and $g_k^*$ are covered by the T-graphs $g(\Gamma_{h(k)-1})$ and $g(\Gamma_{h(k)})$, respectively. Denote by $C_k^+$ and $C_k^-$ the closed communicating classes in $g(\Gamma_{h(k)-1})$ containing the sinks $s_k^+$ and $s_k^-$, respectively, then $C_k^+ \subset S_k^+$ and $C_k^- \subset S_k^-$, and the sinks $s_k^+$ and $s_k^-$ are the unique main states of $C_k^+$ and $C_k^-$, respectively.
Theorem 3.4.1. Under Assumption 2.2.2, let $\mu_{C_k^+}$ and $\mu_{C_k^-}$ be the metastable distributions on $C_k^+$ and $C_k^-$ (Eq. (3.5)), respectively. Then $\tilde{\psi}_k$ and $\tilde{\varphi}_k$ defined by

$$
\tilde{\psi}_k(i) = \begin{cases} 
\mu_{C_k^+}(i), & i \in C_k^+, \\
-\mu_{C_k^-}(i), & i \in C_k^-, \\
0, & i \in S \setminus (C_k^+ \cup C_k^-),
\end{cases}
\quad
\tilde{\varphi}_k(i) = \begin{cases} 
1, & i \in S_k^+, \\
0, & i \in S \setminus S_k^+,
\end{cases}
$$

for $k = 1, \cdots, n-1$, are asymptotic estimates of the left and right eigenvectors $\psi_k$ and $\varphi_k$ in the sense that there exist $\rho_1, \rho_2 > 0$ such that

(i) $\tilde{\psi}_k^T \tilde{\varphi}_{k'} = 1_{(k=k')} + o(\exp(-\rho_1/\varepsilon))$;

(ii) (a) $\tilde{\psi}_k^T L \tilde{\varphi}_k = -\lambda_k (1 + o(\exp(-\rho_2/\varepsilon)))$;

(b) If $k > k'$, then $\lim_{\varepsilon \to 0} \varepsilon \ln(|\tilde{\psi}_k^T L \tilde{\varphi}_{k'}|) < -\Delta_k$;

(c) If $k < k'$, then $\lim_{\varepsilon \to 0} \varepsilon \ln(|\tilde{\psi}_k^T L \tilde{\varphi}_{k'}|) \leq -\Delta_{k'}$.

The proof of Theorem 3.4.1 is provided in Appendix D, which makes use of the weak nested property of optimal W-graphs (Theorem 2.3.1).

3.5 A Non-Recursive Implementation Scheme for Constructing T-graphs

Although conceptually, the recursive structure of Algorithm 2 resonates with the hierarchical structure of T-graphs, practically, it is neither convenient nor efficient for implementation. In this section, we describe a non-recursive implementation scheme which is essentially object-oriented. Note that we had proposed a non-recursive implementation scheme for constructing optimal W-graphs (Algorithm 1)
in [8]. There it was used to obtain sharp asymptotic estimates of eigenvalues and trace in-trees. However, the implementation scheme in [8] was not object-oriented.

The hierarchy of T-graphs can be thought of as a collection of typical “portraits” of a metastable system on each characteristic timescale. As the system jumps from one phase to the next, its “portrait” advances from one T-graph to the next. Two critical observations can be made regarding the evolution of T-graphs: (1) arcs are only allowed to grow out of closed communicating classes (i.e. each newly grown arc should originate from a closed communicating class and head to a vertex outside of this closed communicating class); (2) the growth of each new arc will either result in a path leading to a different closed communicating class (exit), or result in a larger closed communicating class containing the origin closed communicating class (loop).

Furthermore, let us think of T-graphs as being composed of three types of objects: vertices, communicating classes, and arcs. As the system evolves, the statuses of these objects change accordingly, achieving compatibility with the global structure of the current T-graph. So what should the statuses encode, so that by keeping track of them we will be able to grow T-graphs efficiently?

A vertex should monitor which communicating class it currently belongs to, how much weight it currently carries, and the set of recurrent main states (termini) a process starting from this vertex can reach on current timescale.

A communicating class is constructed on its birth timescale. It remains closed until the timescale on which it collapses (or is exited). Accordingly, its birth timescale and exit rate should be recorded, as well as how it is exited. In ad-
dition, it should record the set of member vertices, the corresponding metastable
distribution, and the main states.

A candidate arc \((i \rightarrow j)\) considered to be added to the T-graphs is first screened
for qualifications in accordance with observation (1) above, namely, \(i\) and \(j\) do not
communicate and \(i\) currently belongs to a closed communicating class. If qualified,
it will compete with other qualified arcs in achieving minimum effective weight. A
convenient data structure for such a selection is a min-heap. Once selected, this arc
will either be an exit arc or lead to a loop. In the first case, the communicating
class containing the tail \(i\) collapses, its exit rate and exit path are recorded, and
all its member vertices have to update their set of termini. In the second case, a
larger communicating class is born and its birth timescale is recorded. A breath-
first-search can be used to find its member vertices, and simultaneously, update the
weights of these vertices as well as which communicating class they currently belong
to.

Computation costs lie mostly in maintaining the min-heap, constructing new
communicating classes, and updating vertex weights. We present the pseudocodes
in Appendix E.
Chapter 4: Applications

4.1 Biasing the Random Walk of a Biomolecular Motor

In this section, we apply the concept of T-graphs to analyze the random walk of a molecular motor. The example considered is based on Astumian’s model [1].

Bimolecular motors are molecules that convert chemical free energy (often provided by ATP hydrolysis) into directed motion and into the performance of work on the environment. The sequence of conformational changes of a molecule can be described as a random walk. According to its chemical state, the free energy landscape of a motor fluctuates. The transition rates from state $i$ to state $j$ on each free energy landscape are of the form

$$L_{ij} = A \exp \left( -\frac{U_{ij}}{k_BT} \right),$$

where $F_i$ and $F_{ij}$ are the free energy at state $i$ and the barrier separating $i$ and $j$, respectively, $k_B$ is Boltzmann constant, $T$ is the absolute temperature, and $A$ is a pre-factor. Here $U_{ij} = F_{ij} - F_i$ and $\varepsilon = k_BT$.

At chemical and thermal equilibrium, the corresponding Markov jump process is reversible and there is no biased motion. However, when the chemical reaction (ATP hydrolysis) is out of equilibrium due to the excess of ATP, the corresponding
Markov jump process is no longer reversible and the motor will move preferentially in one direction.

Now let us look at a specific example. Kinesin is a biomolecular motor moving on a microtubule that is polar with two distinguishable ends, “front” and “back”. Experimental evidence indicates that kinesin moves on its track in a “walking” motion where the two heads move hand-over-hand. In a very simple picture, we can talk about four states: $1 = \{\text{right head front, left head back}\}$, $2 = \{\text{right head attached, left head free}\}$, $3 = \{\text{right head back, left head front}\}$, $4 = \{\text{right head free, left head attached}\}$. Cycling through the states in the order $1 \to 2 \to 3 \to 4 \to 1$ leads to a step forward, while cycling in the order $1 \to 4 \to 3 \to 2 \to 1$ leads to a step backward.

Consider an external control parameter $\psi(t)$ which is identified with the effect of ATP hydrolysis. In Astumian’s model [1], $\psi(t)$ fluctuates between two values, $+\Psi$ and $-\Psi$. Correspondingly, the free energy landscape switches between two shapes, shown as red and blue curves in Figure 4.1. When $\psi(t) = +\Psi$, state 2 is globally most stable; when $\psi(t) = -\Psi$, it is state 4.

To distinguish between the two chemical states, we double the state space to $\{1_+, 2_+, 3_+, 4_+, 1_-, 2_-, 3_-, 4_-\}$. The resulting eight-state random walk is represented as the graph $G$ in Figure 4.2. The arc weights are synthetic, made up based on the shape of the energy landscapes. The transition rates between $+\Psi$ and $-\Psi$ are of the order of $\exp(-\zeta/\varepsilon)$.

When $\zeta$ is tuned to the interval $(6, 9.5)$, the corresponding hierarchy of T-graphs is shown in Figure 4.2. It predicts that the motor will most likely accomplish
Figure 4.1: The time-dependent free energy landscape of Kinesin. The forward motion, shown with the arrows, occurs for a certain range of values of $\zeta$, where the switching rate between $+\Psi$ and $-\Psi$ is of the order of $\exp(-\zeta/\varepsilon)$. (Courtesy of Professor Maria Cameron.)

the first forward motion on the timescale of the order of $\exp(\zeta/\varepsilon)$, as is shown in the T-graph $g(\Gamma_5)$, and the typical walking style is $4_+ \rightleftharpoons 1_+ \rightarrow 2_+ \rightarrow 2_- \rightleftharpoons 3_- \rightarrow 4_- \rightarrow 4_+$, which suggests frequent oscillations between $4_+$ and $1_+$ as well as between $2_-$ and $3_-$ before relaxations to $2_+$ and $4_-$, respectively.

Figure 4.3 presents the T-graphs on the timescales when the Kinesin motor is most likely to accomplish the first forward motion for various ranges of values of $\zeta$. When $\zeta$ lies in these three intervals, $(4.5, 5)$, $(5, 5.5)$, and $(5.5, 6)$, the Kinesin motor is expected to accomplish the first forward motion on the timescale of order of $\exp(6/\varepsilon)$, which is the smallest timescale among all ranges of values of $\zeta$. On one hand, smaller values of $\zeta$ (less than 4.5) will lead to frequent switching between the two chemical states, which hinders transitions within each free energy landscape. On
the other hand, larger values of $\zeta$ (greater than 6) will impede transitions between the two chemical states, and the motor is confined within one free energy landscape for a long period of time. Note that in Figure 4.3, although the T-graphs $g(\Gamma_5)$ when $\zeta \in (5, 5, 5)$ and when $\zeta \in (5.5, 6)$ look the same, they are actually different. When $\zeta \in (5, 5, 5)$, the oscillations $4+ \rightleftharpoons 4-$ and $2- \rightleftharpoons 2+$ happen before the transitions $4+ \rightarrow 1+$ and $2- \rightarrow 3-$, however, when $\zeta \in (5.5, 6)$, it is the other way around.
Figure 4.2: The hierarchy of T-graphs of the eight-state random walk model of the Kinesin motor when $\zeta \in (6, 9.5)$. Blue and yellow vertices correspond to recurrent and transient states, respectively.
Figure 4.3: The T-graphs on the timescales when the Kinesin motor is most likely to accomplish the first forward motion for various ranges of values of $\zeta$. Brown arcs represent the slowest transitions. Blue and yellow vertices correspond to recurrent and transient states, respectively.
4.2 Asymptotic Analysis of The LJ\textsubscript{75} Network

In this section, we apply Algorithm 1 (optimal W-graphs) to conduct zero-temperature asymptotic analysis of the stochastic network representing the energy landscape of Lennard-Jones cluster of 75 atoms (LJ\textsubscript{75} network).\textsuperscript{1} Vertices in this network correspond to local potential minima, while arcs represent transition states between them. We will refer to the local minima by their indices in the full dataset of 593, 320 minima provided by D. Wales. The energy landscape of LJ\textsubscript{75} has a double-funnel structure \cite{25} (Figure 4.4). The global minimum, a Marks decahedron with the point group \textit{D}_5\textit{h}, lies at the bottom of the deep and narrow funnel. The second lowest minimum, an icosahedrally packed non-symmetric structure (its point group is \textit{C}_1), locates at the wide and shallow funnel.

The process of physical interest is the transition from the second lowest minimum to the global minimum. The indices of these minima are 92 and 1 respectively in Wales’s dataset. Algorithm 1 allows us to (1) extract the exponential factor of the associated exit rate (since exit rates are asymptotically equivalent to the eigenvalues), and (2) identify the most likely zero-temperature transition path by tracing the unique directed path from minimum 92 to minimum 1 in the associated optimal W-graph.

The LJ\textsubscript{75} network is reversible. The pairwise transition rates in this network can be modeled using a harmonic approximation \cite{26}. In this case, the off-diagonal

\textsuperscript{1}The data for the LJ\textsubscript{75} network were kindly provided by Professor D. Wales, Cambridge University, UK.
Figure 4.4: The disconnectivity graph of the energy landscape of LJ$_{75}$. (Courtesy of Professor David Wales.)
entries of the generator matrix are

\[ L_{ij} = \frac{O_i(\Pi_+\nu_i)^{1/2}}{O_{ij}(\Pi_+\nu_{ij})^{1/2}} \exp \left( -\frac{V_{ij} - V_i}{T} \right), \quad (4.2) \]

where \( O_i \) and \( O_{ij} \) are the point group orders of the local minimum \( i \) and the transition state \((ij)\) separating the local minima \( i \) and \( j \) respectively, while \( \Pi_+\nu_i \) and \( \Pi_+\nu_{ij} \) are the products of the positive eigenvalues of the Hessian matrices of the potential \( V \) at the minimum \( i \) and the transition state \((ij)\) respectively, and \( V_i \) and \( V_{ij} \) are the values of the potential at \( i \) and \((ij)\) respectively. Here \( U_{ij} = V_{ij} - V_i \) and \( \epsilon = T \).

The LJ\(_{75}\) dataset contains 593, 320 local minima and 452, 315 transition states. Wales’s network of LJ\(_{75}\) is disconnected as some saddles are not found. We will restrict our attention to the largest connected component containing the two deepest potential minima, 1 and 92, which consists of 169, 523 vertices and 226, 377 transition states (the number of arcs is twice as large).

The collection of exponential factors \( \Delta_k \) extracted by Algorithm 1 for the LJ\(_{75}\) network is shown is Figure 4.5. As one can observe, there are no notable gaps separating these exponential factors. The exponential factor corresponding to the escape process from the icosahedral funnel to the decahedron funnel is extracted at \( k = 4395 \), where the collapsing sink is \( s^{+}_{4395} = 92 \) and the absorbing sink is \( s^{-}_{4395} = 1 \). The pre-factor and increment found by Algorithm 1 are \( \alpha_{4395} = 147.2 \) and \( \Delta_{4395} = 7.897 \), so the estimated transition rate from minimum 92 to minimum 1 for very low temperature is

\[ \lambda_{4395} \approx 147.2 \exp(-7.897/T). \quad (4.3) \]

The bottleneck transition \((p_{4395} \rightarrow q_{4395})\) is found to be arc \((25811 \rightarrow 73992)\). The
Figure 4.5: The exponential factors $\Delta_k$ for the LJ$_{75}$ network, where $k = 1, \cdots, 169522$ and the eigenvalues $\lambda_k$ of the generator matrix are of the order of $\exp(-\Delta_k/\varepsilon)$. (Courtesy of Professor Maria Cameron.)

The full transition path predicted by the optimal W-graph $g^*_4395$ is presented in Figure 4.6. The in-tree $S^+_{4395}$ contains 92883 vertices and the closed communicating class $C^+_{4395}$ contains 28032 vertices.

A more detailed study on spectral analysis and clustering of the LJ$_{75}$ network can be found in [8].
Figure 4.6: The energy along the asymptotic zero-temperature path (the MinMax path) connecting the two lowest minima 1 and 92 of LJ$_{75}$. The energy at the global minimum 1 is assumed to be zero. The numbers next to each minimum are their indices in D. Wales’s dataset. (Courtesy of Professor Maria Cameron.)
Chapter 5: Conclusion

In this dissertation, we proposed a graph-algorithmic approach to the study of metastable behavior in Markov jump processes with exponentially small transition rates. In particular, we propose efficient algorithms to compute the asymptotic estimates of eigenvalues and eigenvectors via constructing optimal W-graphs (Algorithm 1) and T-graphs (Algorithm 2).

We started from Wentzell’s asymptotic estimates of eigenvalues using optimal W-graphs. Under the assumption that all optimal W-graphs are unique (Assumption 2.2.2), first, we obtained refined asymptotic estimates of eigenvalues when pre-factors of transition rates are known (Theorem 2.2.3). Secondly, we established a weak nested property of optimal W-graphs (Theorem 2.3.1). Thirdly, we proposed a greedy/dynamical programming Algorithm 1 for constructing the hierarchy of optimal W-graphs as well as extracting the exponential factors of the asymptotic estimates of eigenvalues. It can be used to obtain refined asymptotic estimates of eigenvalues if the pre-factors of the transition rates are given.

To handle the case with symmetry, we introduced the hierarchy of T-graphs and the associated sequence of characteristic timescales, which are defined inductively via Algorithm 2. The hierarchy of T-graphs unifies Wentzell’s hierarchy of
optimal W-graphs and Freidlin’s hierarchy of Markov chains. We showed that closed communicating classes in T-graphs can be seen as metastable classes between two characteristic timescales, and estimates of metastable distributions can be calculated using vertex weights of the respective T-graphs (section 3.3). Under the assumption that all optimal W-graphs are unique, we derived asymptotic estimates of eigenvectors (Theorem 3.4.1). An object-oriented, non-recursive implementation scheme for constructing the hierarchy of T-graphs was also proposed (section 3.5).

We applied the proposed algorithms to systems arising from chemical physics. First, we applied the concept of T-graphs to investigate how an external control parameter can bias the random walk of the molecular motor Kinesin. Secondly, we applied Algorithm 1 to conduct zero-temperature asymptotic analysis of the LJ$_{75}$ network. In particular, it allowed us to predict the most likely zero-temperature transition path from the second lowest minimum to the global minimum and estimate the transition rate.

In the near future, we hope to explore applications to other complex real-world systems exhibiting metastability. One possible application area is in finance, in particular, pricing options with long maturity using eigenfunction expansion [9,11]. Choosing the state space and estimating the transition rates from available data are two critical steps in setting up the model and require further study.
Appendix A: Proof of Theorem 2.2.3

The following notations will be used throughout the proof.

- The two nonnegative integers $k$ and $l$ constitute a conjugate pair in the sense that $k + l = n$.

- $S^l = \{i_1i_2\cdots i_l \mid \text{distinct } i_m \in [n] \text{ for } m = 1, \ldots, l\}$, where $[n] = \{1, 2, \ldots, n\}$.

- $O^l = \{i_1i_2\cdots i_l \in S^l \mid i_1 < i_2 < \cdots < i_l\}$.

- The permutation $\sigma : S^l \to O^l$ maps every sequence $i_1i_2\cdots i_l \in S^l$ to an order sequence in $O^l$. Denote by $|\sigma(i_1, i_2, \cdots, i_l)|$ the number of inversions in $\sigma$. The map $\sigma$ is onto but not one to one.

- Associated with the permutation map $\sigma$, there is an equivalent relationship between two sequence in $S^l$: $i_1i_2\cdots i_l \sim j_1j_2\cdots j_l$ if and only if $\sigma(i_1i_2\cdots i_l) = \sigma(j_1j_2\cdots j_l)$. Hence, $O^l = S^l/\sim$.

- For any $i_1\cdots i_l \in O^l$, denote by $L_{i_1\cdots i_l}$ the submatrix of $L$ such that the set of row and column indices that remain is $\{i_1, \ldots, i_l\}$.

- $\mathcal{H}(l)$ is the collection of subgraphs of $G(S, A, U)$ consisting of spanning subgraphs with exactly $l$ arcs (no self-loops) emanated from distinct vertices. $\mathcal{G}(k)$
is the collection of all possible W-graphs with \( k \) sinks, and \( \mathcal{G}(k) \subsetneq \mathcal{H}(l) \).

- For any \( i_1 \cdots i_l \in \mathcal{O}^l \), denote by \( \mathcal{H}(\{i_1, \cdots, i_l\}) \) the sub-collection of \( \mathcal{H}(l) \) consisting of spanning subgraphs with exactly \( l \) arcs (no self-loops) emanated from each of the vertices \( \{i_1, \cdots, i_l\} \). The sub-collections \( \mathcal{H}(\{i_1, \cdots, i_l\}) \) are disjoint for different sequences \( i_1 \cdots i_l \in \mathcal{O}^l \), and

\[
\mathcal{H}(l) = \bigcup_{i_1 \cdots i_l \in \mathcal{O}^l} \mathcal{H}(\{i_1, \cdots, i_l\}). \tag{A.1}
\]

**Proof.** The proof of Theorem 2.2.3 consists of the following three steps.

Step 1: The characteristic polynomial of the generator matrix \( L \) is

\[
P_L(t) = \det(tI - L) = t^n + \sum_{l=1}^{n-1} C_l t^{n-l}, \tag{A.2}
\]

where the coefficients \( C_l \) are given by

\[
C_l = \sum_{i_1 \cdots i_l \in \mathcal{O}^l} \det(-L_{i_1 \cdots i_l})
= (-1)^l \sum_{i_1 \cdots i_l \in \mathcal{O}^l} \sum_{j_1 \cdots j_l \sim i_1 \cdots i_l} (-1)^{\sigma(j_1, \cdots, j_l)} L_{i_1j_1} \cdots L_{i_lj_l}. \tag{A.3}
\]

Step 2: Evoking the zero row sum property of the generator matrix \( L \), we further simplify Eq.(A.3) to the compact expression

\[
C_l = \sum_{g \in \mathcal{G}(k)} \Pi(g), \text{ where } \Pi(g) = \prod_{(i \rightarrow j) \in g} L_{ij}. \tag{A.4}
\]

The derivation of Eq.(A.4) is further divided into two steps:

Step 2.1:

\[
C_l = (-1)^l \sum_{g \in \mathcal{H}(l)} a_g \Pi(g); \tag{A.5}
\]
Step 2.2:

\[\begin{cases} \ (-1)^l, & \text{if } g \in G(k), \\ 0, & \text{if } g \in H(l) \setminus G(k). \end{cases}\]  \hspace{1cm} (A.6)

Step 3: Comparing coefficients in two different forms of writing the characteristic polynomial,

\[P_L(t) = t^n + \sum_{l=1}^{n-1} C_l t^{n-l} = t(t + \lambda_1) \cdots (t + \lambda_{n-1}),\]  \hspace{1cm} (A.7)

we obtain the following estimates of eigenvalues,

\[\lambda_k = \frac{\Pi(g_k^*)}{\Pi(g_{k+1}^*)} (1 + o(\exp(-\rho/\varepsilon))).\]  \hspace{1cm} (A.8)

Eq. (A.8) is precisely Eq. (2.3).

Now we present each step in detail.

Step 1:

Consider instead the following polynomial with \(n\) variables \(t_1, t_2, \ldots, t_n\),

\[P_L(t_1, t_2, \ldots, t_n) = \det(\text{diag}\{t_1, t_2, \ldots, t_n\} - L)\]

\[= t_1 \cdots t_n + \sum_{l=1}^{n-1} \sum_{i_1, \ldots, i_l \in \mathcal{O}^l} C_{i_1 \cdots i_l} t_{i_{l+1}} \cdots t_{i_l},\]  \hspace{1cm} (A.9)

where \(\{i_{l+1}, \ldots, i_n\} = \{1, \ldots, n\} \setminus \{i_1, \ldots, i_l\}\). Now replace all \(t_i\) by \(t\), we recover \(P_L(t)\) and the coefficients

\[C_t = \sum_{i_1, \ldots, i_l \in \mathcal{O}^l} C_{i_1 \cdots i_l}.\]  \hspace{1cm} (A.10)
To compute $C_{i_1 \cdots i_l}$, write

$$P_L(t_1, \cdots, t_n) = \left| \begin{array}{ccc} t_{i_1} - L_{i_1i_1} & \cdots & \ast \\ \vdots & \ddots & \vdots \\ \ast & \cdots & t_{i_l} - L_{i_li_l} \\ t_{i_{l+1}} - L_{i_{l+1}i_{l+1}} & \cdots & \vdots \\ & & t_{i_n} - L_{i_ni_n} \end{array} \right| = P_{L_{i_1 \cdots i_l}}(t_{i_1}, \cdots, t_{i_l}) P_{L_{i_{l+1} \cdots i_n}}(t_{i_{l+1}}, \cdots, t_{i_n}) + \text{constant.} \quad (A.11)$$

It is clear from Eq. (A.11) that

$$C_{i_1 \cdots i_l} = P_{L_{i_1 \cdots i_l}}(0, \cdots, 0) = \det (-L_{i_1 \cdots i_l}). \quad (A.12)$$

Eq. (A.10) and (A.12), together with the Leibniz formula give Eq. (A.3).

Step 2:

Setp 2.1:

Consider each product term $L_{i_{j_1} \cdots i_{j_l}}$ in Eq. (A.3), where $i_1 \cdots i_l \in \mathcal{O}^l$, $i_1 \cdots i_l \sim j_1 \cdots j_l$. Suppose $i_1 \cdots i_l$ and $j_1 \cdots j_l$ agree on exactly $s$ entries, say

$$i_{m_1} = j_{m_1}, \cdots, i_{m_s} = j_{m_s}, \text{ (self-loops)} \quad (A.13)$$

$$i_{m+1} \neq j_{m+1}, \cdots, i_{m_l} \neq j_{m_l}, \text{ and } i_{m+1} \cdots i_{m_l} \sim j_{m+1} \cdots j_{m_l} \in \mathcal{S}^{l-s}. \quad (A.14)$$

Note that $s$ can take any integer from 0 to $l$ except for $l - 1$. 

67
If \( s \leq l - 2 \), the set of arcs \( \{(i_{m+1} \rightarrow j_{m+1}), \ldots, (i_m \rightarrow j_m)\} \) along with the subset of vertices \( \{i_{m+1}, \ldots, i_m\} \) constitute a subgraph in which every vertex has exactly one incoming and one outgoing arc. Such a subgraph can only be made up of cycles of length no smaller than two.

Now bring in the zero row sum property of the generator matrix \( L \) to replace any diagonal entries (i.e. \( i_m = j_m \)) appear in the product term. Specifically,

\[
L_{i_1 j_1} \cdots L_{i_l j_l} = L_{i_1 m_1} \cdots L_{i_1 m_1} L_{i_2 m_2} L_{i_2 m_2 + 1} \cdots L_{i_2 m_2 + 1} \cdots L_{i_l m_l} L_{i_l m_l + 1} \cdots L_{i_l m_l + 1} L_{i_l m_l} \cdots L_{i_l m_l} \cdot (A.15)
\]

Graphically, we break each self-loop at \( i_m \), then add the subset of arcs in \( A \) originated from \( i_m \).

For each product term in Eq.(A.15), the associated set of arcs \( \{(i_{m_1} \rightarrow d_1), \ldots, (i_{m_s} \rightarrow d_s), (i_{m_s + 1} \rightarrow j_{m_s + 1}), \ldots, (i_m \rightarrow j_m)\} \) constitutes an spanning subgraph belonging to \( H(\{i_1, \ldots, i_l\}) \), and it contains at least cycles composed by the subset of arcs \( \{(i_{m+1} \rightarrow j_{m+1}), \ldots, (i_m \rightarrow j_m)\} \). Easy to observe that

\[
\sum_{j_1 \ldots j_l \sim i_1 \cdots i_l} (-1)^{\sigma(j_1, \ldots, j_l)} L_{i_1 j_1} \cdots L_{i_l j_l} = \sum_{g \in H(\{i_1, \ldots, i_l\})} a_g \Pi(g), \quad (A.16)
\]

for some scalers \( a_g \).

Eq.(A.1), (A.3) and (A.16) together give Eq.(A.5). Next we calculate the scalers \( a_g \),

68
Step 2.2:

(1) If \( g \in G(k) \), then \( g \) contains no cycles, \( \Pi(g) \) can only come from the product term \( L_{i_1 i_1} \cdots L_{i_l i_l} \) (i.e. \( s = l \)) after breaking all self-loops. Since

\[
L_{i_1 i_1} \cdots L_{i_l i_l} = \left( - \sum_{d_1 \neq i_1} L_{i_1 d_1} \right) \cdots \left( - \sum_{d_l \neq i_l} L_{i_l d_l} \right) = (-1)^l \sum_{d_1 \neq i_1, \ldots, d_l \neq i_l} L_{i_1 d_1} \cdots L_{i_l d_l}, \tag{A.17}
\]

we must have \( a_g = (-1)^l \) for all \( g \in G(k) \).

(2) If \( g \in H(l) \setminus G(k) \), then \( g \) contains at least one cycle of length no smaller than two. Suppose \( g \) contains \( h \geq 1 \) cycles, \( c^t = \{(i^t_1 \rightarrow j^t_1), \ldots, (i^t_m \rightarrow j^t_m)\}, \ t = 1, \ldots, h \), we can write

\[
\Pi(g) = \Pi(g \setminus \cup_{t=1}^h c^t) \prod_{t=1}^h \Pi(c^t). \tag{A.18}
\]

Each \( \Pi(c^t) \) can come from product terms containing \( L_{i^t_1 j^t_1} \cdots L_{i^t_m j^t_m} \) or \( L_{i^t_1 i^t_1} \cdots L_{i^t_m i^t_m} \) (after breaking all self-loops). In general, \( \Pi(g) \) can come from product terms of the following form,

\[
(L_{i_1 i_1} \cdots L_{i_l i_l}) \frac{\left( L_{i^1_1 j^1_1} \cdots L_{i^1_m j^1_m} \right)^{\delta_1}}{\left( L_{i^1_1 i^1_1} \cdots L_{i^1_m i^1_m} \right)^{\delta_1}} \cdots \frac{\left( L_{i^h_1 j^h_1} \cdots L_{i^h_m j^h_m} \right)^{\delta_h}}{\left( L_{i^h_1 i^h_1} \cdots L_{i^h_m i^h_m} \right)^{\delta_h}}, \tag{A.19}
\]

where \( \delta_t \in \{0, 1\}, t = 1, \ldots, h \). To proceed, we need the following lemma.

**Lemma A.0.1.** Let \( \{j_1, \ldots, j_l\} = \{1, \ldots, l\}, l \geq 2 \). Suppose \( \{(1 \rightarrow j_1), \ldots, (l \rightarrow j_l)\} \) forms a cycle of length \( l \), then \( |\sigma(j_1 \cdots j_l)| = l - 1 \).

**Proof.** Prove by induction. For \( l = 2 \), the only possible situation is \( j_1 j_2 = 21 \), and clearly \( |\sigma(21)| = 1 \). Assume the lemma is true for \( l = k \geq 2 \), consider
l = k + 1 ≥ 3. Suppose along the cycle \( c = \{(1 \rightarrow j_1), \cdots, (k + 1, \rightarrow j_{k+1})\} \), 1 is the next to \( i_1 \), i.e., \( (i_1 \rightarrow 1) \in c \). Since \( l \geq 3 \), \( i_1 \neq j_1 \). Now in the sequence \( j_1 \cdots j_l \), swap \( j_1 \) and 1. Graphically, we redirect the arc \( (i_1 \rightarrow 1) \) to \( (i_1 \rightarrow j_1) \) and the arc \( (1 \rightarrow j_1) \) to the self-loop \( (1 \rightarrow 1) \). As a result, we obtain the set of arcs \( D = \{(1 \rightarrow 1), (2 \rightarrow j_2), \cdots, (i_1 \rightarrow j_1), \cdots, (k + 1, \rightarrow j_{k+1})\} \), and \( D \setminus \{(1 \rightarrow 1)\} \) forms a cycle of length \( k \geq 2 \). Therefore, by induction \( |\sigma(j_1 \cdots j_{k+1})| = 1 + (k - 1) = k \).

Now continue with Eq.(A.19), one can deduce that

\[
a_g = \sum_{\delta_t, \, t=1,\cdots,h} (-1)^{l-m_1} \cdots -m_h \delta_h \prod (-1)^{l} \sigma(j_{m_1}^1 \cdots j_{m_h}^h) \delta_h \delta_t \delta_1 \cdots \delta_h = \sum_{\delta_t, \, t=1,\cdots,h} (-1)^{l-m_1} \cdots -m_h \delta_h \prod (-1)^{l-m_1} \delta_1 \cdots (-1)^{m_h-1} \delta_h = \sum_{\delta_t, \, t=1,\cdots,h} (-1)^{l-\delta_1} \cdots -\delta_h = \sum_{\delta_t, \, t=2,\cdots,h} (-1)^{l-\delta_2} \cdots -\delta_h (-1)^0 + (-1)^{-1} = 0,
\]

i.e. \( a_g = 0 \) for all \( g \in H(l) \setminus G(k) \).

Step 3:

Since the roots of the characteristic polynomial \( P_L(t) = \det(tI - L) \) are eigenvalues of \( L \), we can also write

\[
P_L(t) = t(t + \lambda_1) \cdots (t + \lambda_{n-1}),
\]
and easy to see

\[ C_l = \sum_{i_1 \cdots i_l \in O^l} \lambda_{i_1} \cdots \lambda_{i_l}, \text{ for } l = 1, \cdots, n - 1. \] (A.22)

Compare Eq. (A.4) and Eq. (A.22) we obtain

\[ \frac{C_l}{C_{l-1}} = \frac{\sum_{i_1 \cdots i_l \in O^l} \lambda_{i_1} \cdots \lambda_{i_l}}{\sum_{i_1 \cdots i_{l-1} \in O^{l-1}} \lambda_{i_1} \cdots \lambda_{i_{l-1}}} = \frac{\Pi_{G(k)} \Pi(g)}{\Pi_{G(k+1)} \Pi(g)}. \] (A.23)

Since \( L_{ij} \) are of the form \( \kappa_{ij} \exp(-U_{ij}/\varepsilon) \), all summations in Eq. (A.23) are dominated by their respective slowest decaying terms as \( \varepsilon \to 0 \). According to Assumption 2.2.2, all optimal W-graphs are unique, so all \( \lambda_k \)'s are real and distinct for \( \varepsilon \) sufficiently small. Hence for each \( k \), there exits positive \( \rho_1 \) and \( \rho_2 \) such that

\[ \frac{C_l}{C_{l-1}} = \lambda_k (1 + o(\exp(-\rho_1/\varepsilon))) = \frac{\Pi(g^*_{k})}{\Pi(g^*_{k+1})} (1 + o(\exp(-\rho_2/\varepsilon))). \] (A.24)

Since there are only finitely many \( \lambda_k \), there exits \( \rho > 0 \) such that

\[ \lambda_k = \frac{\Pi(g^*_{k})}{\Pi(g^*_{k+1})} (1 + o(\exp(-\rho/\varepsilon))), \text{ for all } k = 1, \cdots, n - 1. \] (A.25)

\[ \square \]
Appendix B: Proof of Theorem 2.3.1

Proof. (i) Since $g_k^*$ has only $k$ sinks, there must exists an in-tree of $g_{k+1}^*$ whose sink is not a sink of $g_k^*$. Let us denote this in-tree by $S_k^+$ and its sink by $s_k^+$. Uniqueness is the consequence of (ii).

(ii) Suppose $A(g_k^*; S \setminus S_k^+) \neq A(g_{k+1}^*; S \setminus S_k^+)$. Since $|A(g_k^*; S \setminus S_k^+)| = |A(g_{k+1}^*; S \setminus S_k^+)| = n - |S_k^+| - k$, and in $g_{k+1}^*$ there is no arc between $S_k^+$ and $S \setminus S_k^+$, we can swap the subset of arcs $A(g_{k+1}^*; S \setminus S_k^+)$ and $A(g_k^*; S \setminus S_k^+)$ of $g_{k+1}^*$ and $g_k^*$, respectively, without creating any cycle, thus resulting in the following two W-graphs,

\begin{align*}
\hat{g}_{k+1} &= A(g_{k+1}^*; S_k^+) \cup A(g_k^*; S \setminus S_k^+), \\
\hat{g}_k &= A(g_k^*; S_k^+) \cup A(g_{k+1}^*; S \setminus S_k^+). \tag{B.1}
\end{align*}

From our assumption, $\hat{g}_{k+1} \neq g_{k+1}^*$ and $\hat{g}_k \neq g_k^*$. Then the unique optimality of $g_{k+1}^*$ and $g_k^*$ commands that

\begin{align*}
\mathcal{V}(\hat{g}_{k+1}) > \mathcal{V}(g_{k+1}^*) &\implies \mathcal{V}(A(g_{k+1}^*; S \setminus S_k^+)) > \mathcal{V}(A(g_{k+1}^*; S \setminus S_k^+)), \tag{B.3} \\
\mathcal{V}(\hat{g}_k) > \mathcal{V}(g_k^*) &\implies \mathcal{V}(A(g_k^*; S \setminus S_k^+)) > \mathcal{V}(A(g_k^*; S \setminus S_k^+)). \tag{B.4}
\end{align*}

However, Eq.(B.3) and Eq.(B.4) are in contradiction with each other.
(iii) Suppose in $g_k^*$ there are $l \geq 2$ arcs from $S_k^+$ to $S \setminus S_k^+$. Then the subset of vertices $S_k^+$ can be divided into $l$ disjoint subsets $X_1, \ldots, X_l$, such that the induced subgraph $g_k^*|_{X_h}$ is an in-tree and there is a single arc from $X_h$ to $S \setminus S_k^+$ for all $h = 1, \ldots, l$. Without loss of generality, let $s_k^+$ belong to $X_1$, then $|A(g_{k+1}; S_k^+ \setminus X_1)| = |A(g_k^*; S_k^+ \setminus X_1)| = |S_k^+| - |X_1|$, since $S_k^+ \setminus X_1$ includes neither sinks of $g_{k+1}^*$ nor sinks of $g_k^*$. Furthermore, in $g_{k+1}^*$ there is no arc between $S \setminus S_k^+$ and $S_k^+ \setminus X_1$, and in $g_k^*$ there is no arc between $S_k^+ \setminus X_1$ and $X_1$. Thus, we can swap the subset of arcs $A(g_{k+1}; S_k^+ \setminus X_1)$ and $A(g_k^*; S_k^+ \setminus X_1)$ of $g_{k+1}^*$ and $g_k^*$, respectively, creating no cycle and resulting in the following two W-graphs,

$$
\bar{g}_{k+1} = A(g_{k+1}; S \setminus (S_k^+ \setminus X_1)) \cup A(g_k^*; S_k^+ \setminus X_1), \quad (B.5)
$$

$$
\bar{g}_k = A(g_k^*; S \setminus (S_k^+ \setminus X_1)) \cup A(g_{k+1}^*; S_k^+ \setminus X_1). \quad (B.6)
$$

Since $l \geq 2$, $A(g_{k+1}; S_k^+ \setminus X_1) \neq A(g_k^*; S_k^+ \setminus X_1)$, so $\bar{g}_{k+1} \neq g_{k+1}^*$ and $\bar{g}_k \neq g_k^*$. Then the unique optimality of $g_{k+1}^*$ and $g_k^*$ commands that

$$
\mathcal{V}(\bar{g}_{k+1}) > \mathcal{V}(g_{k+1}^*) \implies \mathcal{V}(A(g_k^*; S_k^+ \setminus X_1)) > \mathcal{V}(A(g_{k+1}; S_k^+ \setminus X_1)), \quad (B.7)
$$

$$
\mathcal{V}(\bar{g}_k) > \mathcal{V}(g_k^*) \implies \mathcal{V}(A(g_{k+1}; S_k^+ \setminus X_1)) > \mathcal{V}(A(g_k^*; S_k^+ \setminus X_1)). \quad (B.8)
$$

However, Eq. (B.7) and Eq. (B.8) are in contradiction with each other.

\[ \square \]
Appendix C: Proof of Theorem 2.5.2

The following notations will be used throughout the proof.

- For a vertex $i^{(r)} \in S^{(r)}$, denote by $(i^{(r)} \rightarrow J(i^{(r)}))$, its unique min-outgoing arc in $G^{(r)}$.

- $c^{(r)}$ is the cycle encountered in the $r$th recursion of Algorithm 1. Since all the arcs of $c^{(r)}$ are min-outgoing arcs in $G^{(r)}$, we will identify $c^{(r)}$ with its set of vertices.

- $y^{(r)}$ the tail of the maximum weight arc in the cycle $c^{(r)}$.

- Denote by $\tilde{g}_k^{(r)}$ the graphs produced by Algorithm 1 in the $r$th recursion, where $r = 0, \cdots, \hat{r}$ and $k = 1, \cdots, k_r - 1$.

It is clear from the construction of Algorithm 1 that $\tilde{g}_k^{(r)} \in G^{(r)}(k)$, i.e. they are W-graphs of $G^{(r)}$, also the hierarchy of $\{\tilde{g}_k^{(r)}\}$ satisfies the weak nested property.

To show that indeed $\tilde{g}_k^{(r)} = g_k^{*(r)}$ for all $r$ and $k$, it is suffice to prove the following two lemmas.

**Lemma C.0.2.** If $\tilde{g}_{k-1}^{(r+1)} = g_{k-1}^{*(r+1)}$, $\tilde{g}_{k}^{(r+1)} = g_{k}^{*(r+1)}$ and $\tilde{g}_{k-1}^{(r)} = g_{k-1}^{*(r)}$, then $\tilde{g}_k^{(r)} = g_k^{*(r)}$, where $0 \leq r \leq \hat{r} - 1$ and $2 \leq k \leq k_{r+1} - 1$. 
Lemma C.0.3. If $\tilde{g}_1^{(r+1)} = g_1^{s(r+1)}$, then $\tilde{g}_1^{(r)} = g_1^{s(r)}$, where $0 \leq r \leq \hat{r} - 1$.

At the bottom of the recursion, $\tilde{g}_k^{(\hat{r})} = g_k^{s(\hat{r})}$ trivially for $k = 1, \cdots, k_\hat{r} - 1$.

From the bottom, let inductions proceed in the reverse direction of the construction procedure of Algorithm 1. Specifically, from $r = \hat{r} - 1$ to $r = 0$ and from $k = 1$ to $k = k_{r+1} - 1$, apply the above two lemmas in a chain of inductions on $\tilde{g}_k^{(r)}$ to conclude that $\tilde{g}_k^{(r)} = g_k^{s(r)}$. For $k = k_r - 1, \cdots, k_{r+1}$, by construction $\tilde{g}_k^{(r)} = g_k^{s(r)}$ trivially.

To prove Lemma C.0.2, we first need to justify the following Claim C.0.4.

Claim C.0.4. Under the assumptions of Lemma C.0.2, the subset of vertices in $c^{(r)}$ is covered by one in-tree of the optimal W-graph $g_k^{s(r)}$, where $k = 2, \cdots, k_{r+1} - 1$.

Proof of Claim C.0.4: Suppose $c^{(r)}$ is covered by more than one in-trees of $g_k^{s(r)}$.

Since by construction $c^{(r)}$ is covered by one in-tree of $\tilde{g}_k^{(r)}$, and by assumption $\tilde{g}_k^{(r)} = g_k^{s(r)}$, then $c^{(r)}$ is covered by one in-tree of $g_k^{s(r)}$. From the weak nested property of optimal W-graphs, we infer that $c^{(r)}$ intersects exactly two in-trees of $g_k^{s(r)}$. Denote these two in-trees by $T_1^{(r)}$ and $T_2^{(r)}$ respectively, such that $T_1^{(r)}$ is to be absorbed by $T_2^{(r)}$ when forming $g_k^{s(r)}$, then $A^{(r)}(g_k^{s(r)}; T_2^{(r)}) = A^{(r)}(g_k^{s(r)}, T_2^{(r)})$.

There exist an arc $(u^{(r)} \rightarrow J(u^{(r)}))$ of $c^{(r)}$, with $u^{(r)} \in T_1^{(r)}$ and $J(u^{(r)}) \in T_2^{(r)}$, such that $(u^{(r)} \rightarrow J(u^{(r)})) \in g_{k-1}^{s(r)}$ but $(u^{(r)} \rightarrow J(u^{(r)})) \notin g_k^{s(r)}$.

1. If $u^{(r)}$ is not a sink of $g_k^{s(r)}$, then the in-tree $T_1^{(r)}$ of $g_k^{s(r)}$ contains an arc $(u^{(r)} \rightarrow v^{(r)})$ and $U_{u^{(r)}}^{(r)} > U_{u^{(r)}, J(u^{(r)})}^{(r)}$. However, replacing the arc $(u^{(r)} \rightarrow v^{(r)})$ by $(u^{(r)} \rightarrow J(u^{(r)}))$ creates no cycles while the total weight decreases, a contradiction to the unique optimality of $g_k^{s(r)}$. 

75
(2) If \( u^{(r)} \) is a sink of \( g^{s(r)}_k \), then \( u^{(r)} \) is the sink of the in-tree \( T^{(r)}_1 \). First look at the in-tree \( T^{(r)}_2 \) of \( g^{s(r)}_k \), find a lowest vertex \( p^{(r)} \in c^{(r)} \cap T^{(r)}_2 \), meaning, from \( p^{(r)} \) the unique directed path leading to the sink of \( T^{(r)}_2 \) passes no other vertices belonging to \( c^{(r)} \), so the arc \( (p^{(r)} \rightarrow J(p^{(r)})) \notin g^{s(r)}_k \). (At this point, we do not know yet whether such a lowest vertex exists uniquely.) Because of the weak nested property, \( p^{(r)} \) is also a lowest vertex among \( c^{(r)} \) in \( g^{s(r)}_{k-1} \), or equivalently \( g^{s(r)}_{k-1} \), then we know from construction that except for the arc \( (p^{(r)} \rightarrow J(p^{(r)})) \), all arcs in \( c^{(r)} \) belong to \( g^{s(r)}_{k-1} \), or equivalently \( g^{s(r)}_{k-1} \). Since \( A^{(r)}(g^{s(r)}_{k}; T^{(r)}_2) = A^{(r)}(g^{s(r)}_{k-1}; T^{(r)}_2) \), we conclude that for any vertex \( i^{(r)} \in c^{(r)} \cap T^{(r)}_2 \) such that \( i^{(r)} \neq p^{(r)} \), the min-outgoing arc \( (i^{(r)} \rightarrow J(i^{(r)})) \) belongs to the in-tree \( T^{(r)}_2 \). So \( p^{(r)} \) is indeed the lowest and the in-tree \( T^{(r)}_2 \) contains the section from \( J(u^{(r)}) \) to \( p^{(r)} \) of the cycle \( c^{(r)} \).

Next look at the in-tree \( T^{(r)}_1 \), for any vertex \( i^{(r)} \in c^{(r)} \cap T^{(r)}_1 \) such that \( i^{(r)} \neq u^{(r)} \), the min-outgoing arc \( (i^{(r)} \rightarrow J(i^{(r)})) \) must belong to the in-tree \( T^{(r)}_1 \). Otherwise, we can replace all arcs of \( g^{s(r)}_k \) originated from such \( i^{(r)} \) by their min-outgoing arcs \( (i^{(r)} \rightarrow J(i^{(r)})) \) without creating any cycle while the total weight decreases. To see why there is no cycle resulting from such an operation, if there were any cycle formed, it must pass some redirected arcs, more precisely, a section of the cycle \( c^{(r)} \), which will inevitably lead to the sink \( u^{(r)} \). Indeed, the in-tree \( T^{(r)}_1 \) covers the section from \( J(p^{(r)}) \) to \( u^{(r)} \) of the cycle \( c^{(r)} \).

To sum up, (i) except for the two arcs \( (p^{(r)} \rightarrow J(p^{(r)})) \) and \( (u^{(r)} \rightarrow J(u^{(r)})) \), all arcs in \( c^{(r)} \) belong to \( g^{s(r)}_k \); (ii) except for the arc \( (p^{(r)} \rightarrow J(p^{(r)})) \), all arcs
in \( c^{(r)} \) belong to \( g_{k-1}^*(r) \), or equivalently \( \tilde{g}_{k-1}^{(r)} \).

(a) If \( p^{(r)} \) is not a sink of \( g_k^*(r) \), then the in-tree \( T_2^{(r)} \) of \( g_k^*(r) \) contains an arc \( (p^{(r)} \rightarrow z^{(r)}) \) and \( U_{p^{(r)}z^{(r)}} > U_{p^{(r)}J(p^{(r)})} \). However, replacing the arc \( (p^{(r)} \rightarrow z^{(r)}) \) by \( (p^{(r)} \rightarrow J(p^{(r)}) \) creates no cycles while the total weight decreases, a contradiction to the unique optimality of \( g_k^*(r) \).

(b) If \( p^{(r)} \) is a sink of \( g_k^*(r) \), then \( p^{(r)} \) is the sink of the in-tree \( T_2^{(r)} \). Due to the weak nested property, \( p^{(r)} \) is also a sink of \( g_{k-1}^*(r) \), or equivalently \( \tilde{g}_{k-1}^{(r)} \), then from construction, we know that \( p^{(r)} \) is a sink of \( \tilde{g}_k^{(r)} \) and \( p^{(r)} = y^{(r)} \) must be the case.

Adding to \( g_k^*(r) \) the arc \( (u^{(r)} \rightarrow J(u^{(r)}) \), we obtain a W-graph \( \hat{g}_{k-1}^{(r)} \in G^{(r)}(k-1) \), and

\[
A^{(r)}(\hat{g}_{k-1}^{(r)}; c^{(r)}) = A^{(r)}(g_{k-1}^*(r); c^{(r)}) = A^{(r)}(\tilde{g}_{k-1}^{(r)}; c^{(r)}). \tag{C.1}
\]

A unique W-graph \( \hat{g}_{k-1}^{(r+1)} \in G^{(r)}(k-1) \) can be found such that \( \hat{g}_{k-1}^{(r)} = \text{Expand}_W(\hat{g}_{k-1}^{(r+1)} , c^{(r)}) \). Note that \( \hat{g}_{k-1}^{(r)} = g_{k-1}^*(r) = \tilde{g}_{k-1}^{(r)} \) is possible, correspondingly, \( \hat{g}_{k-1}^{(r+1)} = g_{k-1}^*(r+1) = \tilde{g}_{k-1}^{(r+1)} \). From construction and the optimality of \( g_{k-1}^*(r) \) we deduce that

\[
\mathcal{V}
\left(A^{(r)}(\hat{g}_{k-1}^{(r)}; S^{(r)} \setminus c^{(r)})\right) = \mathcal{V}(\hat{g}_{k-1}^{(r+1)})
\geq \mathcal{V}(g_{k-1}^*(r+1))
= \mathcal{V}(\hat{g}_{k-1}^{(r+1)}) = \mathcal{V}
\left(A^{(r)}(\hat{g}_{k-1}^{(r)}; S^{(r)} \setminus c^{(r)})\right).
\tag{C.2}
\]
Since both $\tilde{g}_{k-1}^{(r)}$ and $\tilde{g}_k^{(r)}$ are not obtained trivially from $G^{(r)}$, the construction procedures assure us that

$$\mathcal{V}(\tilde{g}_{k-1}^{(r)}) - \mathcal{V}(\tilde{g}_k^{(r)}) = \mathcal{V}(\tilde{g}_{k-1}^{(r+1)}) - \mathcal{V}(\tilde{g}_k^{(r+1)}) > U_{y^{(r)}J(y^{(r)})}^{(r)} \geq U_{u^{(r)}J(u^{(r)})}^{(r)}.$$  

(C.3)

However, Eq. (C.1), (C.2) and (C.3) together lead to

$$\mathcal{V}(g_k^{*(r)}) = \mathcal{V}(\tilde{g}_{k-1}^{(r)}) - U_{u^{(r)}J(u^{(r)})}^{(r)}$$

$$= \mathcal{V} \left( A^{(r)}(\tilde{g}_{k-1}^{(r)}; S^{(r)} \backslash c^{(r)}) \right) + \mathcal{V} \left( A^{(r)}(\tilde{g}_{k-1}^{(r)}; c^{(r)}) \right) - U_{u^{(r)}J(u^{(r)})}^{(r)}$$

$$\geq \mathcal{V} \left( A^{(r)}(\tilde{g}_{k-1}^{(r)}; S^{(r)} \backslash c^{(r)}) \right) + \mathcal{V} \left( A^{(r)}(\tilde{g}_{k-1}^{(r)}; c^{(r)}) \right) - \left( \mathcal{V}(\tilde{g}_{k-1}^{(r)}) - \mathcal{V}(\tilde{g}_k^{(r)}) \right)$$

$$= \mathcal{V}(\tilde{g}_{k-1}^{(r)}) - \left( \mathcal{V}(\tilde{g}_{k-1}^{(r)}) - \mathcal{V}(\tilde{g}_k^{(r)}) \right)$$

$$= \mathcal{V}(\tilde{g}_k^{(r)}),$$

(C.4)

a contradiction to the unique optimality of $g_k^{*(r)}$.

Now we are ready to prove Lemma C.0.2.

**Proof of Lemma C.0.2:** First look at the W-graph $\tilde{g}_k^{(r)}$. It is clear from construction that $c^{(r)}$ is covered by one in-tree of $\tilde{g}_k^{(r)}$, for $k = 2, \cdots, k_r + 1 - 1$, and a unique lowest vertex $\tilde{p}^{(r)} \in c^{(r)}$ exits such that except for the arc $(\tilde{p}^{(r)} \rightarrow J(\tilde{p}^{(r)}))$, all arcs in $c^{(r)}$ belong to $\tilde{g}_k^{(r)}$.

1. If $\tilde{p}^{(r)}$ is not a sink of $\tilde{g}_k^{(r)}$, then $\tilde{g}_k^{(r)}$ contains an arc $(\tilde{p}^{(r)} \rightarrow \tilde{z}^{(r)})$ such that

$$U_{\tilde{p}^{(r)}\tilde{z}^{(r)}}^{(r)} > U_{\tilde{p}^{(r)}J(\tilde{p}^{(r)})}^{(r)}.$$
(2) If \( p^{(r)} \) is a sink of \( g_k^{(r)} \), then by construction \( \hat{p}^{(r)} = y^{(r)} \).

Next look at the optimal W-graph \( g_k^{*(r)} \). From Claim C.0.4 we know that \( c^{(r)} \) is covered by one in-tree of \( g_k^{*(r)} \), for \( k = 2, \ldots, k_{r+1} - 1 \). It can be argue similarly as in (2) of the proof of Claim C.0.4 that a unique lowest vertex \( p^{(r)} \in c^{(r)} \) exists such that except for the arc \( (p^{(r)} \rightarrow J(p^{(r)})) \), all arcs in \( c^{(r)} \) belong to \( g_k^{*(r)} \).

(1) If \( p^{(r)} \) is not a sink of \( g_k^{*(r)} \), then \( g_k^{*(r)} \) contains an arc \( (p^{(r)} \rightarrow z^{(r)}) \) such that \( U_{p^{(r)}z^{(r)}}^{(r)} > U_{p^{(r)}J(p^{(r)})}^{(r)} \).

(2) If \( p^{(r)} \) is a sink of \( g_k^{*(r)} \), then we must have \( p^{(r)} = y^{(r)} \). Otherwise, replacing the arc \( (y^{(r)} \rightarrow J(y^{(r)})) \) by \( (p^{(r)} \rightarrow J(p^{(r)})) \) creates no cycles while the total weight decreases, a contradiction to the unique optimality of \( g_k^{*(r)} \).

A unique W-graph \( g_k^{*(r+1)} \in \mathcal{G}^{(r+1)}(k) \) can be found such that \( g_k^{*(r)} = \text{Expand}_W(\hat{g}_k^{(r+1)}, c^{(r)}) \).

Suppose \( g_k^{*(r)} \neq \hat{g}_k^{(r)} \), then \( g_k^{*(r+1)} \neq \hat{g}_k^{(r+1)} \neq \hat{g}_k^{(r+1)} = g_k^{*(r+1)} \). From construction and the unique optimality of \( g_k^{*(r+1)} \) we deduce that

\[
\mathcal{V}(g_k^{*(r+1)}) = \mathcal{V}\left(A^{(r)}(g_k^{*(r)}; S^{(r)} \setminus c^{(r)})\right)
\]

\[
+ \begin{cases} 
U_{p^{(r)}z^{(r)}}^{(r)} + U_{y^{(r)}J(y^{(r)})}^{(r)} - U_{p^{(r)}J(p^{(r)})}^{(r)}, & \text{if } p^{(r)} \text{ is not a sink of } g_k^{*(r)}, \\
0, & \text{if } p^{(r)} = y^{(r)} \text{ is a sink of } g_k^{*(r)},
\end{cases}
\]

\[
> \mathcal{V}(g_k^{*(r+1)}) = \mathcal{V}(\hat{g}_k^{(r+1)}) = \mathcal{V}\left(A^{(r)}(\hat{g}_k^{(r)}; S^{(r)} \setminus c^{(r)})\right)
\]

\[
+ \begin{cases} 
U_{\hat{p}^{(r)}z^{(r)}}^{(r)} + U_{y^{(r)}J(y^{(r)})}^{(r)} - U_{\hat{p}^{(r)}J(\hat{p}^{(r)})}^{(r)}, & \text{if } \hat{p}^{(r)} \text{ is not a sink of } \hat{g}_k^{(r)}, \\
0, & \text{if } \hat{p}^{(r)} = y^{(r)} \text{ is a sink of } \hat{g}_k^{(r)}.
\end{cases}
\]

(C.5)
Now add the constant $\mathcal{V}(c^{(r)}) - U^{(r)}_{y^{(r)}J(y^{(r)})}$ to both side of Eq. (C.5) we obtain

$$\mathcal{V}(g^{s(r)}_{k}) = \mathcal{V}(A^{(r)}(g^{s(r)}_{k}; S^{(r)} \setminus c^{(r)}))$$

$$+ \begin{cases} 
U^{(r)}_{\tilde{p}^{(r)}z^{(r)}} + \mathcal{V}(c^{(r)}) - U^{(r)}_{\tilde{p}^{(r)}J(\tilde{p}^{(r)})}, & \text{if } \tilde{p}^{(r)} \text{ is not a sink of } g^{s(r)}_{k}, \\
\mathcal{V}(c^{(r)}) - U^{(r)}_{y^{(r)}J(y^{(r)})}, & \text{if } \tilde{p}^{(r)} = y^{(r)} \text{ is a sink of } g^{s(r)}_{k}.
\end{cases}$$

$$\mathcal{V}(\tilde{g}^{(r)}_{k}) = \mathcal{V}(A^{(r)}(\tilde{g}^{(r)}_{k}; S^{(r)} \setminus c^{(r)}))$$

$$+ \begin{cases} 
U^{(r)}_{\tilde{p}^{(r)}z^{(r)}} + \mathcal{V}(c^{(r)}) - U^{(r)}_{\tilde{p}^{(r)}J(\tilde{p}^{(r)})}, & \text{if } \tilde{p}^{(r)} \text{ is not a sink of } \tilde{g}^{(r)}_{k}, \\
\mathcal{V}(c^{(r)}) - U^{(r)}_{y^{(r)}J(y^{(r)})}, & \text{if } \tilde{p}^{(r)} = y^{(r)} \text{ is a sink of } \tilde{g}^{(r)}_{k}.
\end{cases}$$

(C.6)

a contradiction to the unique optimality of $g^{s(r)}_{k}$.

\[ \square \]

**Proof of Lemma C.0.3:** The proof essentially follows the same lines as the proof of Lemma C.0.2. In fact, since there is only one in-tree in both $\tilde{g}^{(r)}_{1}$ and $g^{s(r)}_{1}$ (so no need to worry about Claim C.0.4), just replace the index $k$ by 1 in the proof of Lemma C.0.2, everything will go through.

\[ \square \]
Appendix D: Proof of Theorem 3.4.1

Proof. (i) Since the two sinks $s_k^+$ and $s_k^-$ are the unique main states of $C_k^+$ and $C_k^-$, respectively, there exits $\rho_1 > 0$ such that for all $k$

\[
\mu_{C_k^+}(s_k^+) = 1 + o(\exp(-\rho_1/\varepsilon)) \quad (D.1)
\]
\[
\mu_{C_k^+}(i) = o(\exp(-\rho_1/\varepsilon)), \quad i \in C_k^+ \setminus \{s_k^\pm\}. \quad (D.2)
\]

Let

\[
(I) = \tilde{\psi}_k^T \tilde{\varphi}_k = \sum_{i \in C_k^+ \cap S_k^+} \mu_{C_k^+}(i) - \sum_{i \in C_k^- \cap S_k^+} \mu_{C_k^-}(i).
\]

(1) If $k = k'$, then $C_k^+ \cap S_k^+ = C_{k'}^+$ and $C_k^- \cap S_k^+ = \emptyset$, so

\[
(I) = \sum_{i \in C_k^+} \mu_{C_k^+}(i) = 1 + o(\exp(-\rho_1/\varepsilon)). \quad (D.3)
\]

(2) If $k > k'$, then either $S_k^+ \subseteq S_{k'}^+$ or $S_k^+ \cap S_{k'}^+ = \emptyset$ since $S_k^+$ collapses first.

(a) If $S_k^+ \subseteq S_{k'}^+$, then $S_k^- \subseteq S_{k'}^+$ as well, so

\[
(I) = \sum_{i \in C_k^+} \mu_{C_k^+}(i) - \sum_{i \in C_k^-} \mu_{C_k^-}(i) = o(\exp(-\rho_1/\varepsilon)). \quad (D.4)
\]

(b) If $S_k^+ \cap S_{k'}^+ = \emptyset$, then $S_k^- \cap S_{k'}^+ = \emptyset$ as well, so $(I) = 0$.

(3) If $k < k'$, then $S_{k'}^+$ collapses first and any vertex in $S_{k'}^+$ can not be a sink of the optimal W-graph $g_{k+1}^*$, in particular, can not be $s_k^\pm$. Thus

\[
(I) = o(\exp(-\rho_1/\varepsilon)). \quad (D.5)
\]

81
(ii) From the weak nested property, \((p_k \rightarrow q_k)\) is the unique typical exit from \(C^+_k\), there exists \(\rho_2 > 0\) such that for all \(k\), the exit rate from \(C^+_k\) satisfies

\[
e(C^+_k) = \mu_{C^+_k}(p_k)L_{p_kq_k}(1 + o(\exp(-\rho_2/\varepsilon)))
\]

\[
= \lambda_k(1 + o(\exp(-\rho_2/\varepsilon))). \tag{D.6}
\]

Moreover, the exit rate from \(C^-_k\) is exponentially smaller, i.e.

\[
e(C^-_k) \ll e(C^+_k). \tag{D.7}
\]

Let

\[
(II) = \tilde{\psi}_k^T L \tilde{\varphi}_{k'} = \sum_{i \in C^+_k, j \in S^+_k, j \neq i} \mu_{C^+_k}(i)L_{ij}(\tilde{\varphi}_{k'}(j) - \tilde{\varphi}_{k'}(i)) - \sum_{i \in C^-_k, j \in S^+_k, j \neq i} \mu_{C^-_k}(i)L_{ij}(\tilde{\varphi}_{k'}(j) - \tilde{\varphi}_{k'}(i)).
\]

(1) If \(k = k'\), then \(C^+_k \cap S^+_k = C^+_k\) and \(C^-_k \cap S^+_k = \emptyset\), so

\[
(II) = \sum_{i \in C^+_k, j \not\in S^+_k} \mu_{C^+_k}(i)L_{ij}(0 - 1) - \sum_{i \in C^-_k, j \in S^+_k} \mu_{C^-_k}(i)L_{ij}(1 - 0)
\]

\[
= A - B. \tag{D.8}
\]

Since \(p_k \in C^+_k\) and \(q_k \not\in S^+_k\), \(A = \mu_{C^+_k}(p_k)L_{p_kq_k}(1 + o(\exp(-\rho_2/\varepsilon)))\). \(B\) is no greater than the exit rate from \(C^-_k\), i.e. \(B \leq e(C^-_k)\). Thus from Eqs. (D.6), (D.7) and (D.8) we deduce that

\[
(II) = \lambda_k(1 + o(\exp(-\rho_2/\varepsilon))). \tag{D.9}
\]

(2) If \(k > k'\), then \(\Delta_k < \Delta_{k'}\) and either \(S^+_k \not\subseteq S^+_k\) or \(S^+_k \cap S^+_k = \emptyset\) since \(S^+_k\) collapses first. 

82
(a) If $S_k^+ \subsetneq S_{k'}^+$, then $S_k^- \subsetneq S_{k'}^+$ as well, so

\[
(II) = \sum_{\substack{i \in C_k^+ \cap S_{k'}^+ \cap S_k^- \cap S_{k'}^+ \cap S_k^- \cap S_{k'}^- \cap (S_k^+ \cup S_k^-) = \emptyset}} \mu_{C_k^+}(i)L_{ij}(0-1) - \sum_{\substack{i \in C_k^+ \cap S_{k'}^+ \cap S_k^- \cap S_{k'}^- \cap (S_k^+ \cup S_k^-) = \emptyset}} \mu_{C_k^-}(i)L_{ij}(0-1) = A - B. \quad (D.10)
\]

Since $p_k \in C_k^+$ and $q_k \notin S_{k'}^+ \subsetneq S_{k'}^+$, $|A| \ll e(C_k')$. $|B|$ is no greater than the exit rate from $C_k^-$, i.e. $|B| \leq e(C_k^-)$. Thus from Eqs. (D.7) and (D.10) we deduce that

\[
|(II)| \ll e(C_k'^+). \quad (D.11)
\]

(b) If $S_k^+ \cap S_{k'}^+ = \emptyset$, then $S_k^- \cap S_{k'}^+ = \emptyset$ as well, so

\[
(II) = \sum_{\substack{i \in C_k^+ \cap S_{k'}^+ \cap S_k^- \cap S_{k'}^- \cap (S_k^+ \cup S_k^-) = \emptyset}} \mu_{C_k^+}(i)L_{ij}(1-0) - \sum_{\substack{i \in C_k^- \cap S_{k'}^+ \cap S_k^- \cap S_{k'}^- \cap (S_k^+ \cup S_k^-) = \emptyset}} \mu_{C_k^-}(i)L_{ij}(1-0) = A - B. \quad (D.12)
\]

Since $p_k \in C_k^+$ and $q_k \notin S_{k'}^+ \subsetneq S_{k'}^+$, $A \ll e(C_k')$. $B$ is no greater than the exit rate from $C_k^-$, i.e. $B \leq e(C_k^-)$. Thus from Eqs. (D.7) and (D.12) we deduce that

\[
|(II)| \ll e(C_k'^+). \quad (D.13)
\]

Since $e(C_k') \simeq \lambda_k$, $|((II)| \ll e(C_k'^+)$ is equivalent to $\lim_{\varepsilon \to 0} \varepsilon \ln(|\tilde{\psi}_k^T L \tilde{\phi}_k'|) < -\Delta_k$.

(3) If $k < k'$, then $\Delta_k > \Delta_{k'}$ and $S_{k'}^+$ collapses first, so

\[
e(C_k'^+) \ll e(C_{k'}'^+). \quad (D.14)
\]

There are three cases: $S_k^+ \subsetneq S_{k'}^+$, or $S_k^+ \subsetneq S_{k'}^-$, or $S_k^+ \cap (S_k^+ \cup S_k^-) = \emptyset$. 83
(a) If $S_{k'}^+ \subsetneq S_k^+$, then $C_k^- \cap S_{k'}^+ = \emptyset$, so

$$
(II) = \sum_{i \in C_k^+ \cap S_{k'}^e} \mu_{C_k^+}(i)L_{ij}(1-0) + \sum_{i \in C_k^+ \cap S_{k'}^e} \mu_{C_k^+}(i)L_{ij}(0-1)
- \sum_{i \in C_k^- \cap S_{k'}^e} \mu_{C_k^-}(i)L_{ij}(1-0)
- \sum_{i \in C_k^- \cap S_{k'}^e} \mu_{C_k^-}(i)L_{ij}(0-1)
= A + B - C.
$$

(D.15)

Since $p_{k'} \in S_{k'}^+$ and $q_{k'} \notin S_{k'}^+$, $|A|$ is no greater than the exit rate from $C_{k'}^+$, i.e. $|A| \leq e(C_{k'}^+)$. Since $S_{k'}^+ \cap S_k^- = \emptyset$, $B < e(C_k^+)$. Since $C_k^- \cap S_{k'}^+ = \emptyset$, $C$ is no greater than the exit rate from $C_k^-$, i.e. $C \leq e(C_k^-)$. Thus from Eqs. (D.7), (D.14) and (D.15) we deduce that

$$
|(II)| \leq e(C_{k'}^+).
$$

(D.16)

(b) If $S_{k'}^+ \subsetneq S_k^-$, then $C_k^+ \cap S_{k'}^- = \emptyset$, so

$$
(II) = \sum_{i \in C_k^+ \cap S_{k'}^e} \mu_{C_k^+}(i)L_{ij}(1-0)
- \sum_{i \in C_k^- \cap S_{k'}^e} \mu_{C_k^-}(i)L_{ij}(0-1) - \sum_{i \in C_k^- \cap S_{k'}^e} \mu_{C_k^-}(i)L_{ij}(1-0)
= A - B - C.
$$

(D.17)

Since $S_{k'}^+ \subsetneq S_k^-$, $A$ is no greater than the exit rate from $C_k^+$, i.e. $A \leq e(C_k^+)$. Since $p_{k'} \in S_{k'}^+$ and $q_{k'} \notin S_{k'}^+$, $|B|$ is no greater than the exit rate from $C_{k'}^+$, i.e. $|B| \leq e(C_{k'}^+)$. Since $S_{k'}^+$ collapses into its complement, the
reverse flow from $S_{k'}^+ c$ to $S_k^+$ will have exponentially smaller rates, i.e. $C \ll e(C_k^+)$. Thus from Eqs. (D.14) and (D.17) we deduce that

$$|(II)| \preceq e(C_k^+). \quad (D.18)$$

(c) If $S_{k'}^+ \cap (S_k^+ \cup S_k^-) = \emptyset$, then

$$(II) = \sum_{i \in C_k^+} \mu_{C_k^+}(i)L_{ij}(1 - 0) - \sum_{i \in C_k^-} \mu_{C_k^-}(i)L_{ij}(1 - 0)$$

$$= A - B. \quad (D.19)$$

Since $S_{k'}^+ \cap S_k^- = \emptyset$, $A \ll e(C_k^+)$. Since $S_{k'}^+ \cap C_k^- = \emptyset$, $B$ is no greater than the exit rate from $C_k^-$, i.e. $B \leq e(C_k^-)$. Thus from Eqs. (D.7), (D.14) and (D.19) we deduce that

$$|(II)| \ll e(C_k^+). \quad (D.20)$$

In summary, $|(II)| \preceq e(C_k^+)$ if $k < k'$. Since $e(C_k^+ \prec \lambda_{k'}$, $|(II)| \preceq e(C_{k'}^+)$ is equivalent to $\lim_{\varepsilon \to 0} \varepsilon \ln(|\tilde{\psi}_k^T L \tilde{\varphi}_{k'}|) \leq -\Delta_{k'}$.

$\Box$
Appendix E: Pseudocodes of a Non-Recursive Implementation of Algorithm 2

Vertex
{
    $C$;  // the communicating class it currently belongs to
    $\delta$;  // the weight it currently carries
    $ms$;  // the set of recurrent main states (termini) it can lead to on current timescale
};

Communicating Class
{
    $\Gamma$;  // exponential factor of its birth timescale
    $s$; // a representative main state
    $S$;  // member vertices
    $\mu$;  // exponential factors of metastable distribution
    $\theta$;  // equals to 0 if currently it is closed, otherwise equals to 1
    $E$;  // exponential factor of its exit rate
    $\mathcal{C}_0$;  // the collection of communicating classes its exit arcs entering
    $ms$;  // the set of recurrent main states (termini) it can lead to on the timescale it collapses
};

Arc
{
    $i$;  // tail
    $j$;  // head
    $U$;  // weight (static)
    key[1];  // equals to 1 if $i$ and $j$ currently communicate, otherwise equals to 0
    key[2];  // equals to 0 if currently $i.C$ is closed, otherwise equals to 1
    key[3];  // equals to $U + i.\delta$ (effective weight)
    key[4];  // equals to $i.C$
    key[5];  // equals to 1 if $i.ms \subset j.ms$ (loop), otherwise equals to 0 (exit)
};
Algorithm 3: A non-recursive construction of T-graphs

Initialization: $S, A, \Gamma = 0, \delta, h = 0, l = 0, \text{TTlist}[] = \emptyset, m = 0, \text{Clist}[] = \emptyset;

Output: $(g(\Gamma_h))_0^h, \{\Gamma_h\}_0^h$.

1: function obj_Tgraphs( )
2:     while $A \neq \emptyset$ do
3:         $a_0 = A.top()$ \hspace{1cm} \triangleright \text{Copy top}
4:         if $a_0.key[1] + a_0.key[2] > 0$ then break \hspace{1cm} \triangleright \text{No qualified arcs}
5:     else
6:         $\Gamma_{\text{next}} = a_0.key[3]$;
7:         if $\Gamma_{\text{next}} > \Gamma$ then
8:             for $i \in S$ do $\delta(i) = i.\delta$;
9:         end for
10:        $\Gamma_h = \Gamma; g(\Gamma_h) = (l, \delta)$; \hspace{1cm} \triangleright \text{Record a new T-graph}
11:       $h = h + 1; \Gamma = \Gamma_{\text{next}}$;
12:     end if
13:    $C_0 = a_0.key[4]; \text{Loops} = \emptyset; \text{Exits} = \emptyset$;
14:     do
15:        $a = A.pop();$ \hspace{1cm} \triangleright \text{Pop top.}
16:        $l = l + 1; \text{TTlist}[l] = [a.i; a.j; a.U]$;
17:        if $a.key[5] == 0$ then $\text{Exits} \cup = a$;
18:        else $\text{Loops} \cup = a$;
19:     end if
20:    $a_{\text{next}} = A.top();$
21:     while $a_{\text{next}.key[1]} + a_{\text{next}.key[2]} == 0 \& a_{\text{next}.key[3]} == \Gamma \& a_{\text{next}.key[4]} == C_0$
22:        if $\text{Exits} \neq \emptyset$ then
23:            $C_0.\theta = 1; C_0.E = \Gamma;$ \hspace{1cm} \triangleright \text{Collapse}
24:             for $a \in \text{Exits}$ do
25:                 $\text{C}_0.C_\text{to} \cup = a.j.C; \text{C}_0.m_\text{to} \cup = a.j.m$s;
26:             end for
27:             for $i \in C_0.S$ do $i.m$s = $C_0.m_\text{to}$;
28:         end for
29:    end if
30:    if $\text{Loops} \neq \emptyset$ then \hspace{1cm} \triangleright \text{A new communicating class.}
31:        $C_{\text{new}} = \text{GrowNewC}(C_0, \text{Loops}); m = m + 1; \text{Clist}[m] = C_{\text{new}}$;
32:    end if
33:     Update.arc.key();
34: end while
35:     for $i \in S$ do $\delta(i) = i.\delta$;
36: end for \hspace{1cm} \triangleright \text{Record the last T-graph}
37: $\Gamma_h = \Gamma; g(\Gamma_h) = (l, \delta); \hat{h} = h; \text{return} \ (g(\Gamma_h))_0^h, \{\Gamma_h\}_0^h$;
38: end function

87
Input: $C_0, \text{Loops}$
Output: $C_{\text{new}}$

1: function $\text{GrowNewC}(C_0, \text{Loops})$
2: \quad $C_{\text{new}} = \text{new} . C()$; \hspace{2cm} \triangleright \text{Born.}$
3: \quad $C_{\text{new}} . \Gamma = \Gamma; C_{\text{new}} . s = C_0 . s; C_{\text{new}} . \theta = 0; C_{\text{new}} . S = C_0 . S;$
4: \quad for $i \in C_0 . S$ do
5: \hspace{1cm} $i . C = C_{\text{new}}; C_{\text{new}} . \mu (i) = i . \delta;$
6: \quad end for
7: \quad $C_{\text{leak}} = C_0 . C_{\text{to}}; C_{\text{next}} = \emptyset;$
8: \quad for $a \in \text{Loops}$ do
9: \hspace{1cm} $C_{\text{next}} \cup = a . i . C;$
10: \quad end for
11: \quad while $C_{\text{next}} \neq \{C_0\}$ do
12: \hspace{1cm} $C_{\text{next}} = \emptyset;$
13: \hspace{1cm} for $C \in C_{\text{next}}$ do
14: \hspace{2cm} $C_{\text{new}} . S \cup = C . S;$
15: \hspace{2cm} for $i \in C . S$ do
16: \hspace{3cm} $i . C = C_{\text{new}}; i . \delta = i . \delta + \Gamma - C . E; C_{\text{new}} . \mu (i) = i . \delta;$
17: \hspace{2cm} end for
18: \hspace{1cm} for $\tilde{C} \in C . C_{\text{to}}$ do
19: \hspace{2cm} if $C_0 . s \in \tilde{C} . ms$ then $C_{\text{next}} \cup = \tilde{C};$
20: \hspace{2cm} else $C_{\text{leak}} \cup = \tilde{C};$
21: \hspace{2cm} end if
22: \hspace{1cm} end for
23: \hspace{1cm} end for
24: \hspace{1cm} $C_{\text{next}} = C_{\text{next}} ;$
25: \hspace{1cm} end while
26: \quad if $C_{\text{leak}} \neq \emptyset$ then \hspace{2cm} \triangleright \text{Collapse.}$
27: \quad \quad $C_{\text{new}} . \theta = 1; C_{\text{new}} . E = \Gamma; C_{\text{new}} . C_{\text{to}} = C_{\text{leak}} ;$
28: \quad \quad for $C \in C_{\text{leak}}$ do
29: \hspace{2cm} $C_{\text{new}} . ms \cup = C . ms;$
30: \hspace{2cm} end for
31: \quad \quad for $i \in C_{\text{new}} . S$ do
32: \hspace{2cm} $i . ms = C_{\text{new}} . ms;$
33: \hspace{2cm} end for
34: \quad \quad end if
35: \quad return $C_{\text{new}}$
36: end function
1: function Update_arc_key( )
2:   for a ∈ A do
3:     a.key[1] = (a.i.C == a.j.C);
7:     a.key[5] = (a.i.ms ⊂ a.j.ms);
8:   end for
9:   A.heapify(); ▷ According to their key vectors,
10:  end function
Bibliography


