Appendix E

Finding the Shortest and the Fastest Paths

> There are many paths to the top of the mountain,
> but the view is always the same.
>
> Chinese proverb

Techniques for finding the shortest and the fastest pathways can be useful during sampling for pathways, in post-processing and analysis of DPS databases, and in various optimisations of pathway ensembles [8, 10, 192, 290, 291]. Given a digraph representation of a connected database of minima and transition states it is possible to identify the shortest path between any two minima using a breadth-first search algorithm [154], which runs in linear time. The shortest path, however, is not necessarily the fastest. Recently Evans suggested using Dijkstra's algorithm [153] for finding the path with the largest non-recrossing DPS rate [290]. It was pointed out by Carr, however, that the weight function used in Reference [290] is not positively defined, and the use of the Bellman-Ford-Moore (BFM) algorithm [292–294] was suggested instead [291]. Because Dijkstra's algorithm scales better than BFM algorithm (see Figure E.1) we describe below a weight function that enables us to use the Dijkstra's algorithm for finding the fastest path.

As detailed in Section 4.1.1, within a digraph representation of a coarse-grained PES each minimum is represented by a single node and each transition state is represented by two oppositely directed edges. Here we ignore degenerate rearrangements since they

do not affect the rates, and for cases where more than one transition state links the same pair of minima only the one with the fastest forward and backward rates is used. Adopting the non-recrossing DPS rate definition from Equation 4.11 we can associate forward and backward rate constants

$$\begin{aligned} k_{a,b} &= \frac{P_b^{\mathrm{eq}}}{P_B^{\mathrm{eq}}} \frac{k_{a,i_1}\, k_{i_1,i_2} \cdots k_{i_n,b}}{\sum_{\gamma_1} k_{\gamma_1,i_1} \sum_{\gamma_2} k_{\gamma_2,i_2} \cdots \sum_{\gamma_n} k_{\gamma_n,i_n}}, \\ k_{b,a} &= \frac{P_a^{\mathrm{eq}}}{P_A^{\mathrm{eq}}} \frac{k_{b,i_1}\, k_{i_1,i_2} \cdots k_{i_n,a}}{\sum_{\gamma_1} k_{\gamma_1,i_1} \sum_{\gamma_2} k_{\gamma_2,i_2} \cdots \sum_{\gamma_n} k_{\gamma_n,i_n}} \end{aligned} \tag{E.1}$$

with a pathway $\xi = a, i_1, i_2, \ldots, i_n, b$.

For a detailed description of Dijkstra and BFM algorithms we refer the reader to Reference [154]. A suitable choice of edge weight function to be used with either of the shortest path algorithms must be made. We define the weight of each directed edge $\alpha \rightarrow \beta$ as

$$w(\beta, \alpha) = \ln\left(\frac{\sum_\gamma k_{\gamma,\alpha}}{k_{\beta,\alpha}}\right). \tag{E.2}$$

Note that $w(\beta, \alpha) \neq w(\alpha, \beta)$ and $0 \leqslant w < \infty$. Since the weight is non-negative it is possible to apply Dijkstra's algorithm to find the fastest pathway [154].

It makes sense to include only minima with more than one connection when searching for the fastest pathway. In our calculations we first iteratively identified a connected subset of minima with the number of connections greater than one, and then run the above algorithm for this subset.

To find the fastest path connecting minima a and b in the direction $b \rightarrow a$ it is necessary to solve either a single-source shortest paths problem with minimum b chosen as a source, or a single destination shortest paths problem with minimum a chosen as the destination. In the latter case all the edges in the graph need to be reversed, which can be accomplished by simply swapping the arguments to the weight function in Equation E.2. This reversal is possible because our graph is a symmetric digraph, i.e. if there is an edge $a \rightarrow b$ then there is also an edge $b \rightarrow a$.

A useful check for correct implementation of this algorithm is to verify that the weight of the shortest path from minimum b to minimum a, $W(a, b)$, is related to the

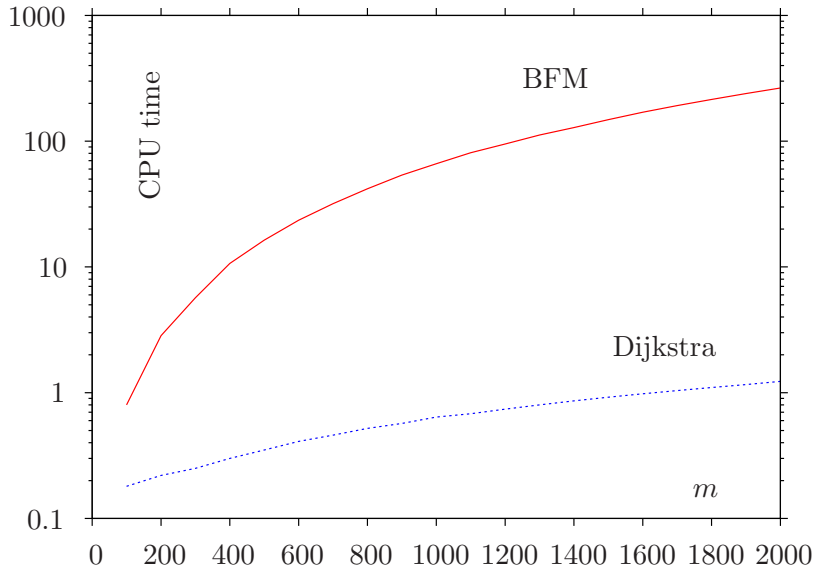

FIGURE E.1: Performance comparison of BFM and static Dijkstra algorithms for an evolving database of minima and transition states for tryptophan zipper 2. CPU time as a function of the number of minima in the database, m, is shown. The number of transition states, t, scales linearly with m, and for the plotted data set the dependence can be approximated as $t = 1.59m - 46.98$ with a correlation coefficient of 1.000. The units of time are arbitrary.

rate $k_{a,b}$ calculated for that pathway by

$$k_{a,b} = \frac{P_b^{\mathrm{eq}}}{P_B^{\mathrm{eq}}} \exp\Big(-W\left(a,b\right)\Big) \sum_{\gamma} k_{\gamma,b}. \tag{E.3}$$

The BFM algorithm runs in $\mathcal{O}(|V|\,|E|)$ time while Dijkstra's algorithm as described in Reference [153] scales as $\mathcal{O}(|V|^2 + |E|) = \mathcal{O}(|V|^2)$ [154]. Although both algorithms appear to have similar (quadratic) asymptotic complexity for the case of sparse graphs (e.g. $|E| = \mathcal{O}(|V|)$) the scaling prefactor is smaller for Dijkstra's algorithm. In addition, for sparse graphs with $|E| = o(|V|^2/\lg|V|)$ an implementation of Dijkstra's algorithm that utilises a priority queue of some sort can improve the asymptotic bounds further. Priority queues based on binary min-heap and Fibonacci heap [276] result in $\mathcal{O}((|E|+|V|)\lg|V|) = \mathcal{O}(|E|\lg|V|)$ and $\mathcal{O}(|V|\lg|V|+|E|)$ scaling of the algorithm, respectively [154]. In the present work we have used a priority queue based on Fibonacci heap. A performance comparison of our implementations of BFM and static Dijkstra's algorithms is shown in Figure E.1.

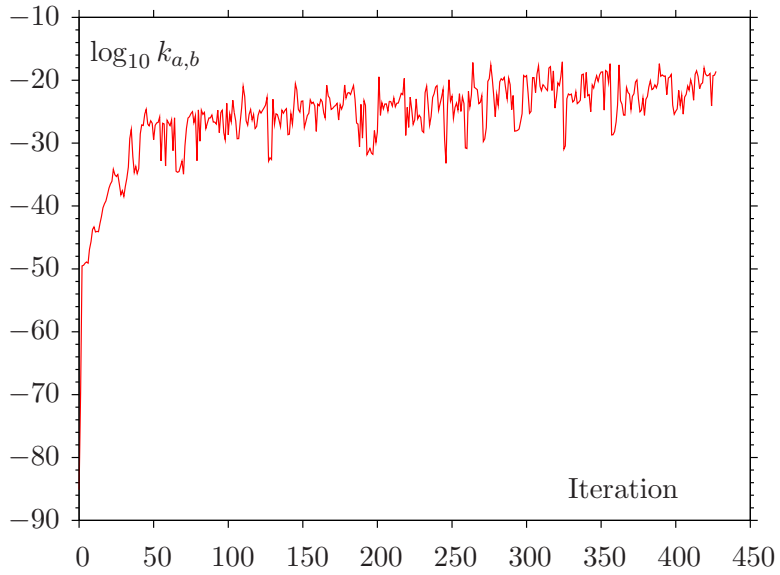

FIGURE E.2: The rate of the fastest pathway, $k_{a,b}$, as a function of the iteration number for the evolving database of tryptophan zipper 3-I. Note $\log_{10}$ scale on vertical axis. As the generation of new connections can affect the rates of the pathways explicitly recorded in the pathway database (see Equation E.1), these rates were recomputed at the end of the calculation. This result, and the fact that $k_{a,b}$ includes recrossing contributions, explains the wiggles in the data. The recrossing contribution was calculated using Algorithm B.1, which is discussed in Chapter 4.

Since knowledge of all the pathways on a complicated PES is out of the question a representative sample must be obtained to properly describe the property of interest. It is unclear what is the best strategy of evolving the pathway database if the proper description of kinetics is sought, as sampling techniques may introduce a bias towards pathways of particular type.

In previous DPS studies a set of perturbations was applied to the fastest known path to generate new ones [8–10]. New pathways were constructed from a subset of stationary points featured in the pathway being perturbed, and from these that were found as a result of perturbation. Because building the shortest paths tree with Dijkstra's algorithm is relatively cheap it is possible to analyse the whole database for the presence of new faster pathways rather than just the subset. Results for an example application are presented in Figure E.2. After a set of perturbations was applied to the fastest currently known pathway $a \leftarrow b$ Dijkstra's algorithm was used to solve the

single-source shortest paths problem to build the shortest paths tree rooted at minimum b. For path ξ the set of perturbations constituted $l(\xi) - \Delta + 1$ attempts to shortcut the path, where Δ is a shortcut parameter, $\Delta \in \{1, 2, \ldots, l(\xi)\}$.

Each attempt to shortcut the path was realised as a single connect run (see Chapter 2) seeded with two minima $\alpha, \beta \in \xi$ separated by a gap of length Δ along the path. If the fastest path $a \leftarrow b$ found by Dijkstra's algorithm was different (not necessarily new) from the one that was currently being perturbed the iteration was completed, this pathway was perturbed in the next iteration and parameter Δ was reset to 1. Otherwise, Δ was incremented by one, unless $\Delta = l(\xi)$ in which case the calculation was terminated.