Supplementary Information

Carl P. Goodrich[a], **Ella M. King**[b, 1], **Samuel S. Schoenholz**[c], **Ekin D. Cubuk**[c], **and Michael Brenner**[a, b, c]

[a]School of Engineering and Applied Sciences, Harvard University, Cambridge, MA 02138, USA; [b]Physics Department, Harvard University, Cambridge, MA 02138, USA; [c]Google Research, 1600 Amphitheatre Pkwy, Mountain View, CA

1. Molecular Dynamics simulations

We use Molecular Dynamics (MD) simulation in two ways. First, in Sec. I, we directly differentiate through MD simulations in order to optimize the honeycomb and triangular lattice assembly rates. Secondly, in Sec. II, we verify our optimized transition rates by running MD simulations and extracting transition rates from the simulations.

The MD simulations are performed using JAX MD (1), a molecular dynamics engine that is compatible with JAX (2), a freely-available automatic differentiation library. The system consists of a two- or three-dimensional box containing N particles at a constant temperature T. To simulate Brownian motion, the dynamics are given by the overdamped Langevin equation:

$$\dot{r}_\alpha = \gamma^{-1} F_\alpha + \sqrt{2 k_B T \gamma^{-1}} f_\alpha(t) \qquad [S1]$$

where F_α is the net force on particle α, $\gamma = 0.1$ is the friction coefficient, k_B is the Boltzmann constant, and the elements of $f_\alpha(t)$ are uncorrelated Gaussian random variables with zero mean.

In the case of optimizing lattice assembly rates, we simulate $N = 100$ particles that interact via Eq. (1). The particles interact within a square two-dimensional simulation box with periodic boundary conditions and sides of length 11.4 and 9.31 for assembling the honeycomb and triangular lattices, respectively. Simulations are performed at a temperature of $k_B T = 0.1$ using a simulation step size of 5×10^{-5}.

In the case of verifying transition rates, we simulate $N = 7$ particles that interact via a Morse potential of the form

$$V_{\alpha\beta}(r_{\alpha\beta}) = B_{\alpha\beta} \left(e^{-2a(r_{\alpha\beta} - \sigma)} - 2e^{-a(r_{\alpha\beta} - \sigma)} \right) \qquad [S2]$$

where $r_{\alpha\beta}$ is the separation between particles α and β, $\sigma = 1$ is the particle diameter, $a = 10$ determines the range of the attraction, and $B_{\alpha\beta}$ is the binding energy between the spheres. Simulations are performed at a temperature of $k_B T = 1$ in a three-dimensional simulation box with free boundary conditions using a step size of 5×10^{-6}.

2. Optimizing lattice assembly rates

A. Forward mode AD. To optimize our system, we use the RMSProp stochastic optimizer (3) with a learning rate of 0.1, a memory value of $\gamma_{\text{mem}} = 0.9$, and a smoothing value of $\epsilon = 10^{-8}$. The optimizer acts on a gradient that is determined via an average of 100 independent simulations, each with random initial positions.

The loss function we use for the optimization consists of specifying a "stencil", or a fragment of a perfect lattice centered around a central particle. The stencil was created with a particle diameter of 1. A honeycomb lattice stencil and a triangular lattice stencil are both specified. At the end of a simulation, we center the stencil on a particle α and define the overlap function

$$O_\alpha(\theta) = \sum_{\beta,\beta'} e^{-\frac{\left(r_\beta - r_{\beta'}(\alpha,\theta)\right)^2}{2\sigma^2}} \qquad [S3]$$

where r_β is the position of particle β and $r_{\beta'}(\alpha, \theta)$ is the position of the β' particle in the stencil when the stencil is centered on particle α and rotated by an angle θ. The maximum overlap of particle α is

$$O_{\alpha,\text{opt}} = \max_\theta O_\alpha(\theta) \qquad [S4]$$

and we define total overlap of the system to be

$$O = \max_\alpha O_{\alpha,\text{opt}}. \qquad [S5]$$

We use a large stencil, ensuring that a significant overlap is clearly indicative of a honeycomb-like region. Additionally, the particles all interact identically. These two features allow us to use the maximum overlap as an indicator of crystallization. The assembly process as measured by the maximum overlap is contrasted with the assembly process as measured by the mean overlap in Fig. S1.

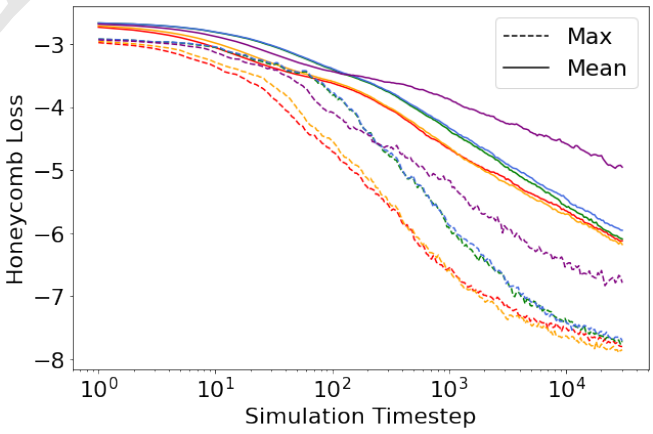

Fig. S1. Assembly process for honeycomb lattices with 5 different sets of parameters, where a given set of parameters is given by a distinct color. The parameters are the same as those used to generate the data in Fig. 2. The assembly process as measured by the maximum over $O_{\alpha,\text{opt}}$ (as shown in S5)is given by dashed lines, whereas the assembly process as measured by the mean over $O_{\alpha,\text{opt}}$ is given by solid lines. We see that the two measurements show the same trend, and both serve as similar indicators of crystallization.

We use a 13 particle stencil for the honeycomb overlap O_{hon}, and 7 particle stencil for the triangular overlap O_{tri}.

[1]Carl P. Goodrich contributed equally to this work with Ella M. King.

These together comprise the two loss functions, weighted as follows:

$$L_H(t) = \frac{1}{N_H}(-O_{\text{hon}}(t) + \xi_H O_{\text{tri}}(t) + \zeta_H) \quad [S6]$$

$$L_T(t) = \frac{1}{N_T}(O_{\text{hon}}(t) - \xi_T O_{\text{tri}}(t) + \zeta_T) \quad [S7]$$

with $\xi_H = 1$ and $\xi_T = \frac{104}{7}$. We choose $\zeta_{H,T}$ and $N_{H,T}$ such that L_H and L_T range between 0 and 1, where $L_{H,T} = 0$ for a perfect lattice.

We perform two rounds of optimization. For input rates $k_H^* = 1/t_H^*$ and $k_T^* = 1/t_T^*$, the first round of optimization runs t^* steps and computes $L = (L_H(t_H^*) - 0.5)^2 + (L_T(t_T^*) - 0.5)^2$ for each of t_H^* and t_T^* under the appropriate density conditions. We then run the simulation for t^* more steps and compute $L_H(2t_H^*)$ and $L_T(2t_T^*)$ at the end of both simulations. The optimization loss function is a sum of both losses, namely (1) how close the system is to half-assembled after t^* steps and (2) how assembled the system is after $2t^*$ steps:

$$L_{\text{opt}} = \sum_{H,T}(L(t^*) - 0.5)^2 + L(2t^*). \quad [S8]$$

The optimal parameters are the parameters associated with the minimum loss value over 1,000 RMSProp optimization steps. We then perform a second optimization, starting with these parameters, in which the optimization loss is restricted to $L_{\text{opt}} = (L_H(t_H^*) - 0.5)^2 + (L_T(t_T^*) - 0.5)^2$. In both rounds of optimization, we use forward mode automatic differentiation to calculate $\frac{dL_{\text{opt}}}{da}$, where a is the set of variable parameters in Eq. (1).

To validate optimization results, we calculate the rates k_{H} and k_{T} by computing the losses $L_H(t)$ and $L_T(t)$ as a function of simulation time step. We then find the earliest timestep, t_0, at which the loss is greater than half its optimal value, and the latest timestep, t_1, at which the loss is less than half its optimal value. Using data in the range $(t_0 - 3, t_1 + 3)$, we approximate a linear fit and use the fit to compute the time t^* at which the loss function is exactly half its optimal value.

This method allows us to gain insight into a highly non-intuitive system by visualizing the results of the optimization. Each computation of the gradient of the potential provides information about the impact of the shape of the potential on the dynamics of the system. In figure S2, we compare our potential after many optimization steps to the final, optimized potential in (4). We also show the initial potential used to start the optimization. We see that our result captures much of the structure of the result found in (4), but rather than finding two local minima in the potential, we observe kinks in the potential in the corresponding locations. This may allow for faster transitions out of non-honeycomb-like metastable states.

B. Reverse mode AD. Performing reverse mode AD is more time efficient, but uses more memory. To conserve memory, we present an indirect approach to optimizing lattice assembly rates in which we only differentiate over the final 300 time steps of the simulation. Consider breaking the simulation up into two components. We first simulate for $\tilde{\tau}$ time steps and find the configuration

$$R_{\tilde{\tau}} = \mathcal{S}_{\text{MD}}\left(a, R_0, \rho, \tilde{\tau}\right) \quad [S9]$$

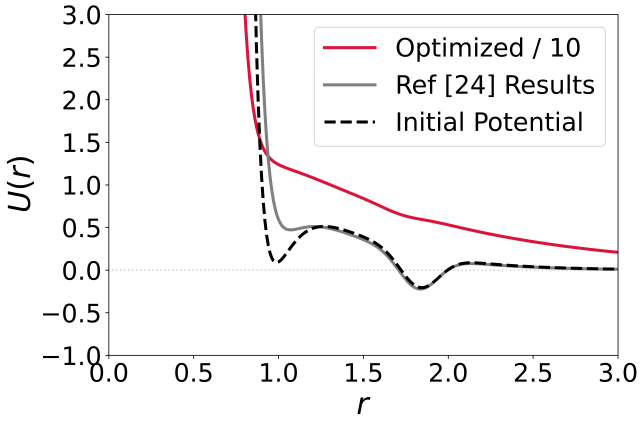

Fig. S2. The two potentials that drive the behavior displayed in figure 1 in the main text are pictured here. The potential that has been optimized to be half-assembled after 3,000 simulation steps is pictured in red, and the final potential found in (4) is pictured in gray. Additionally, the black dashed potential shows the potential associated with the initial parameter values in both our method and in (4). Note that the black dashed potential does not lead to assembly of a honeycomb lattice.

at the end of the first simulation. We then begin from the configuration $R_{\tilde{\tau}}$ and simulate further for $\bar{\tau} = 300$ time steps, returning a final configuration

$$R_{\bar{\tau}} = \mathcal{S}_{\text{MD}}\left(a, R_{\tilde{\tau}}, \rho, \bar{\tau}\right). \quad [S10]$$

Crucially, we differentiate over *only the second simulation.* To optimize the loss at time $\tau = \tilde{\tau} + \bar{\tau}$, we calculate the derivative

$$\left.\frac{d\mathcal{L}\left(\mathcal{S}_{\text{MD}}(a, R_{\tilde{\tau}}, \rho, \bar{\tau})\right)}{da}\right|_{R_{\tilde{\tau}}}. \quad [S11]$$

Calculating this derivative at constant $R_{\tilde{\tau}}$ means that we only have to differentiate through $\bar{\tau}$ time steps. In practice, we find good results with $\bar{\tau}$ as small as 300, which we hold constant while varying $\tilde{\tau}_{\text{H}}$ and $\tilde{\tau}_{\text{T}}$ to tune the two crystallization rates relative to each other. Using our results, we can interpolate and find a relationship between $\tilde{\tau}$ and the corresponding rate: $\tilde{\tau}$ is a knob we can tune to adjust the relative lattice assembly rates.

3. Optimizing transition rates in colloidal clusters

A. The doubly nudged elastic band calculation. To calculate the transition state between two known adjacent local minima in a high-dimensional energy landscape, we follow the procedure from Trygubenko and Wales (5), which we briefly summarize here. We want to find the (monotonically increasing) steepest ascent path from the first minimum up to the transition state and the (monotonically decreasing) steepest descent path down to the second minimum.

Let R_0 and R_{n_I+1} be the configurations of the two minima, and we will represent a path between the two as a series of n_I configurations $\{R_1, R_2, ..., R_{n_I}\}$. As an initial guess, we always choose a simple interpolation between the two minima. Importantly, in order for this to be a reasonable guess, the two minima have to be rotated so that they are close to overlapping. The potential energy of the i-th individual configuration in the

path is $U(R_i)$, which we refer to here as the "true potential." Thus, the total true potential of the ensemble is

$$V = \sum_{i=0}^{n_I+1} U(R_i). \quad [S12]$$

In addition, we connect each adjacent configuration with a high-dimensional spring, leading to the following "elastic band" or "spring" potential

$$\tilde{V} = \frac{1}{2} k_{\text{spr}} \sum_{i=1}^{n_I+1} |R_i - R_{i-1}|^2 . \quad [S13]$$

In principle, one wishes to minimize $V_{\text{tot}} \equiv V + \tilde{V}$ over all the n_I intermediate configurations while keeping the two endpoints fixed at their respective minimum. However, interference between the true and spring potential can give rise to "corner-cutting" and "sliding-down problems." To address these problems, we employ a set of adjustments, or "nudges," to the gradient of V_{tot}, as follows.

Nudging. First, we decompose the gradient of each configuration into components that are parallel and perpendicular to the current path. Let $\hat{\boldsymbol{\tau}}_i$ be the unit vector tangent to the path at configuration i, which is defined as follows. If configuration i does not represent a local optimum, meaning exactly one of its neighbors j has a higher energy, $U(R_i) < U(R_j)$, then its tangent vector is

$$\hat{\boldsymbol{\tau}}_i = \frac{(j-i)(R_j - R_i)}{|R_j - R_i|}. \quad [S14]$$

However, if either both or none of its neighbors are at higher energy, then we use

$$\hat{\boldsymbol{\tau}}_i = \frac{R_{i+1} - R_{i-1}}{|R_{i+1} - R_{i-1}|}. \quad [S15]$$

The gradient of the true potential can then be decomposed into

$$\mathbf{g}_i = \mathbf{g}_i^{\parallel} + \mathbf{g}_i^{\perp}, \quad [S16]$$

where

$$\mathbf{g}_i^{\parallel} = (\nabla_i V \cdot \hat{\boldsymbol{\tau}}_i)\, \hat{\boldsymbol{\tau}}_i, \quad [S17]$$

$$\mathbf{g}_i^{\perp} = \nabla_i V - \mathbf{g}_i^{\parallel}. \quad [S18]$$

Similarly, using tildes to denote quantities related to the spring potential,

$$\tilde{\mathbf{g}}_i = \tilde{\mathbf{g}}_i^{\parallel} + \tilde{\mathbf{g}}_i^{\perp}, \quad [S19]$$

where

$$\tilde{\mathbf{g}}_i^{\parallel} = \left(\nabla_i \tilde{V} \cdot \hat{\boldsymbol{\tau}}_i\right) \hat{\boldsymbol{\tau}}_i, \quad [S20]$$

$$\tilde{\mathbf{g}}_i^{\perp} = \nabla_i \tilde{V} - \tilde{\mathbf{g}}_i^{\parallel}. \quad [S21]$$

The nudged elastic band approach is to project out $\mathbf{g}_i^{\parallel}$ and $\tilde{\mathbf{g}}_i^{\perp}$ when minimizing V_{tot}. This removes some but not all of the interference instabilities. The "doubly nudged" approach is to only project out some of the $\tilde{\mathbf{g}}_i^{\perp}$ term, so that

$$\mathbf{g}_i = \mathbf{g}_i^{\perp} + \tilde{\mathbf{g}}_i^{\parallel} + \tilde{\mathbf{g}}_i^{\perp} - \left(\tilde{\mathbf{g}}_i^{\perp} \cdot \hat{\mathbf{g}}_i^{\perp}\right) \hat{\mathbf{g}}_i^{\perp}. \quad [S22]$$

We proceed by minimizing V_{tot} using this nudged gradient. We note that optimizing over such a connected ensemble is especially straightforward in JAX MD because automatic vectorization is natively built in. The result is a sequence of configurations that closely tracks the steepest descent path we are seeking. Furthermore, the image R_t with the highest energy is an approximation of the true saddle point. Note that we do not refine R_t using eigenvector following (6, 7), a practice that is necessary for many applications. While R_t is therefore only an approximation, this seems to be adequate for our purposes.

B. Optimization of transition kinetics. As discussed in the main text, we optimize the transition kinetics by first calculating R_t using the DNEB method, then calculating $\frac{dL}{dB_{\alpha\beta}}$ using backward mode automatic differentiation, where L is the chosen loss function, and finally using this gradient to minimize L.

The optimization is performed using the RMSProp algorithm as implemented in JAX (2, 3). Note that after each step of the optimization, both the height E_t and the position R_t of the saddle point will change slightly. While it is possible to differentiate over the entire DNEB calculation, this is not necessary and we instead calculate $\frac{dL}{dB_{\alpha\beta}}$ at fixed R_i, R_j, and R_t. We furthermore find it unnecessary in practice to redo the DNEB calculation every optimization step. Instead, we take multiple optimization steps in between DNEB calculations, which increases the efficiency of the computation. We find that recalculating R_t every 50 optimization steps works well for this problem. Note that in Fig. 3B, the iteration number refers to the number of times R_t has been recalculated.

We run a total of 18 such iterations. As before, we use a memory value of $\gamma_{\text{mem}} = 0.9$, and a smoothing value of $\epsilon = 10^{-8}$. However, we use a variable learning rate of 0.064, 0.016, and 0.004 for the first, second, and third set of 6 iterations, respectively.

Finally, we note that due to the long-range tail in the potential (Eq. (S2)), the exact position of the two minima technically change slightly during optimization. Therefore, before calculating R_t each iteration, we first recalculate the local minima, though in practice this does not make a significant difference.

C. Validation of transition rates using MD. To validate the transition rates, we run 100 simulations (described above) for 3×10^6 time steps each. Every 10^4 steps, we compare the positions of the particles to the n_I images, calculated using the DNEB procedure, that compose the transition path between the two states. Specifically, we find the image i that minimizes $\sum_{\alpha\beta}(r_{\alpha\beta} - r_{\alpha\beta,i})^2$, where $r_{\alpha\beta}$ is the distance between particles α and β in the current state, and $r_{\alpha\beta,i}$ is the distance between the particles in image i.

The grey signal in Fig. 3C shows i_{closest} as a function of time. Note that the images near 0 and 100 (roughly 0-20 and 80-100, see flat lines in Fig. 3B) are identical up to rotations, so the distinction between them is meaningless and we do not need to be concerned that the data in Fig. 3C does not reach $i_{\text{closest}} = 0$. Note also that this signal is quite noisy, due in part to fluctuations in directions that do not align with the transition path. Therefore, it is important to filter this signal to remove transients that do not correspond transitions

between the two states. This is done with a second-order lowpass Butterworth filter (see black curve in Fig. 3C).

This filtered signal is then matched to one of the two minima by comparing it to the image number corresponding to the transition state (purple dashed line in Fig. 3C), leading to a binarized signal (blue curve in Fig. 3C). We then calculate the average dwell time of each state, τ_i and τ_j. The measured rates are then $k_{ij,\mathrm{MD}} = 1/\tau_i$ and $k_{ji,\mathrm{MD}} = 1/\tau_j$.

Figure S3 compares the rates extracted in this way to the rates obtained from the Kramer approximation (Eq. 4) and the target rate.

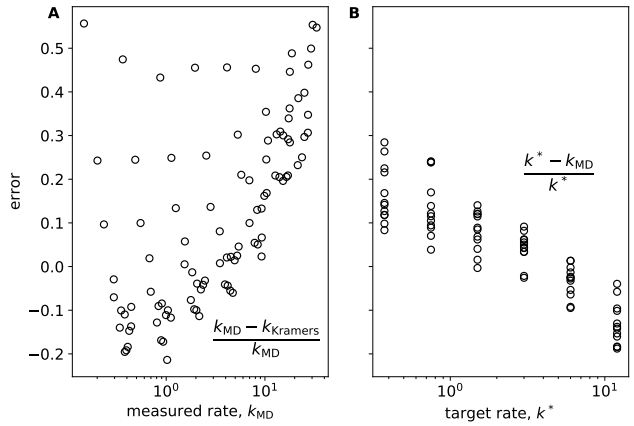

Fig. S3. Comparison of rates. A) Comparison of k_{Kramers} and k_{MD} when targeting desired energy barriers (see Fig. 4B). Perfect agreement is not expected because k_{Kramers} is an approximation that only considers the curvature at two points in the energy landscape, and the observed error of less than 50% is very small compared to the two orders of magnitude of variation in the rates. B) Comparison of k_{MD} and the target rate k^* when targeting desired transition rates (see Fig. 4D). Again, the observed error of mostly less than 20% is expected and very small compared to the variation in the magnitude of the rates. The comparison of k_{Kramers} and k^* is shown in Fig. 4C.

4. Using automatic differentiation in new systems

In determining whether AD can be useful in studying a particular system of interest there are several considerations that must be taken into account. Given a function (that could involve an entire molecular dynamics simulation) that produces a scalar output, reverse-mode AD can compute its gradients using a single pass through the simulation. However, a consequence of this is that the entire simulation trajectory must be retained during the simulation. This induces a memory cost that grows both with the size of the simulation and the number of simulation steps and can quickly become unmanageable. There are several ways of ameliorating this cost. First, one can use gradient rematerialization to recompute short segments of the simulation during the backward pass. This typically reduces the memory cost to scale logarithmically in the duration of the simulation at the cost of a logarithmic increase in the required computational budget. Another option employed here is to use forward-mode AD which does not require additional storage during the simulation. However, here one pass through the entire simulation is required for each parameter and so the computational complexity grows quickly with the number of parameters in the function to be differentiated.

1. SS Schoenholz, ED Cubuk, JAX, M.D.: End-to-End Differentiable, Hardware Accelerated, Molecular Dynamics in Pure Python. *arXiv* (2019).
2. J Bradbury, et al., JAX: composable transformations of Python+NumPy programs. (2018).
3. T Tieleman, G Hinton, Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning* **4**, 26–31 (2012).
4. MC Rechtsman, FH Stillinger, S Torquato, Optimized interactions for targeted self-assembly: application to a honeycomb lattice. *Phys. Rev. Lett.* **95**, 228301 (2005).
5. SA Trygubenko, DJ Wales, A doubly nudged elastic band method for finding transition states. *J. Chem. Phys.* **120**, 2082–2094 (2004).
6. CJ Cerjan, WH Miller, On finding transition states. *J. Chem. Phys.* **75**, 2800–2806 (1981).
7. DJ Wales, Finding saddle points for clusters. *J. Chem. Phys.* **91**, 7002–7010 (1989).