THE JOURNAL OF PHYSICAL CHEMISTRY B

ARTICLE

pubs.acs.org/JPCB

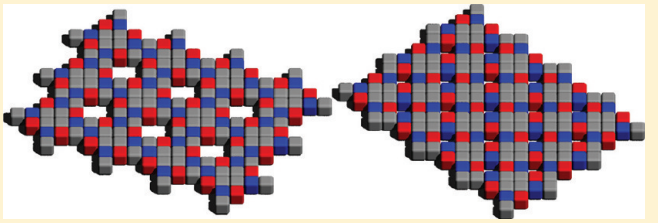
Calculation of Partition Functions for the Self-Assembly of Patchy Particles

Eric Jankowski[†] and Sharon C. Glotzer[*,†,‡]

[†]Department of Chemical Engineering, University of Michigan, Ann Arbor, Michigan 48109, United States

[‡]Department of Materials Science and Engineering, University of Michigan, Ann Arbor, Michigan 48109, United States

ABSTRACT: Partition functions encode all the thermodynamics of a system, but for most systems of practical importance, they cannot be calculated exactly. In this work we present a new hierarchical method for calculating partition functions to arbitrary precision. We discuss the algorithmic details of our implementation, including elements of shape-matching and entropy calculation for on-lattice and off-lattice systems. We highlight computational trade-offs between speed and accuracy, showing how this varies with temperature, and demonstrate its utility in studying nanoscale self-assembly for a system of model patchy particles.



INTRODUCTION

Self-assembly holds promise as a manufacturing technique for materials with customizable physical properties and for devices with precisely placed nanoscale elements.[1,2] Building blocks with complicated shapes and finely tuned directional interactions can now be synthesized from polymers, metals, biomolecules, and combinations of these nanoparticles that have been chemically bound together to create highly anisotropic "patchy" particles.[2−7] The specific interactions and complex shapes of patchy particles can be exploited during self-assembly with particles organizing into complicated patterns as a consequence of free energy minimization.[8] Choosing a building block to assemble a target pattern or assessing whether a system of building blocks might assemble any pattern—the two basic problems of self-assembly—are difficult because of the enormous parameter space that is accessible when particle shape, interaction number, interaction strengths, temperature, density, and stoichiometry can each be varied independently. Methods that can predict what new particles might form and why they do or do not reach their thermodynamically preferred structures are therefore important and useful.

Selecting a building block for which a desired structure is thermodynamically stable would be straightforward if, at experimentally realizable conditions, its partition function could be generated.[9,10] The partition function $\mathcal{Z}$ encodes all the system's thermodynamics, allowing for the calculation of macroscopic properties such as pressure, heat capacity, and packing fraction as a function of thermodynamic variables.[11] Unfortunately, the number of microstates that comprise a particular partition function scales by V^N where V is the volume accessible to each particle and N is the number of particles. This scaling makes exact enumeration of all but the smallest partition functions numerically intractable, and the complicated interactions between real particles renders analytical partition function calculations useless except for Baxter type tricks in special cases.[12,13]

For equilibrium statistical mechanical systems macroscopic properties can be calculated without explicit knowledge of the partition function. This fact follows from the sharp peak in the probability distribution of microstates consistent with equilibrium observables and therefore saddle point approximations, mean field theory, or numerical techniques such as molecular dynamics (MD) and Monte Carlo (MC) methods can be efficiently used to calculate equilibrium properties.[11,14,15] These methods have recently provided integral explanatory and exploratory capabilities for studying self-assembly. Molecular dynamics simulations have been instrumental in explaining the thermodynamics of water,[16] predicting ordered arrangements of diblock copolymers,[17] tethered nanoparticles,[18,19] and anisotropic virus capsomers.[20,21] Monte Carlo simulations have elucidated quasicrystalline structure in systems of hard tetrahedra,[22] solubility of tetromino mixtures,[23] higher-order virial coefficients and clustering in water[24] and explained the dipole-induced structure observed in solutions of stabilized CdTe tetrahedra,[25] among many other examples too numerous to list.

When used as an exploratory tool, the utility of MC and MD simulations is hindered by a circular problem of time scales: to determine the relaxation time for a slowly evolving system, simulations much longer than the relaxation time must be run.[9,26,27] That is, in order to know whether a simulation has been run long enough, one must run it longer. This is a particularly troublesome problem when attempting to find the optimal conditions at which a system of particles self-assembles a target structure because the time scale problem is now

Special Issue: H. Eugene Stanley Festschrift

Received: July 7, 2011
Revised: September 13, 2011
Published: October 17, 2011

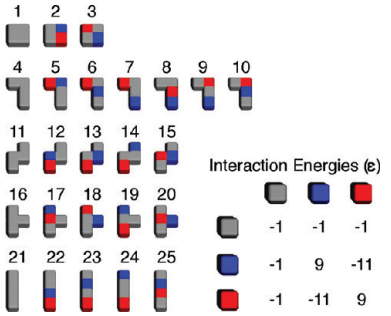


Figure 1. Model patchy particles.

compounded by each additional state point that must be sampled.

In this work we develop a hierarchical method for calculating partition functions for complex building blocks to arbitrary precision by extending bottom-up building block assembly (BUBBA) to all temperatures.[28] Our method exploits the sharply peaked equilibrium distribution of microstates for $\mathcal{Z}$ at low N to hierarchically generate $\mathcal{Z}$ at successively larger values of N. We accomplish this through three steps at each value of N. First, microstates are enumerated through a cluster pairing procedure. Second, distinguishable microstates are identified. Third, negligible microstates are neglected. We demonstrate the method on a system of model patchy particles in the canonical (N,V,T) ensemble.

EXPERIMENTAL METHODS

As a model system, we consider the set of 25 particles introduced by Troisi et al.[29] whose shapes resemble Tetris pieces (tetrominoes, Figure 1[23,30]). For computational expediency and without loss of generality, we confine the rigid, three-dimensional particles to a two-dimensional surface and describe the surface as a square grid in which the subunits that comprise the particles may each occupy a single grid cell. This dimension reduction is general because the method requires only a discrete search space of configurations, which is independent of dimension. In this model,[29] all nearest-neighbor subunits share an attraction (e.g., van der Waals, depletion, solvophobic, etc.) of strength $U = \varepsilon$, and the four subunits that comprise each particle can be positively charged, negatively charged, or neutral. When two like-charged subunits share an edge, their resulting potential energy is $U = 9\varepsilon$, for opposite charges it is $U = -11\varepsilon$, and for a neutral subunit sharing a face with any other subunit type it is $U = -\varepsilon$. The particles can translate and rotate on the grid, but particle overlaps and inversions are prohibited. Since colloids with these geometries and interactions can now be synthesized,[31−33] determining which, if any, of the particles in this family can be used to generate useful structures has immediate applications beyond serving as a proof-of-concept for the methodology presented here.

BUBBA at Finite Temperature. Exact enumeration of a partition function is computationally intractable for large $(N > 5)$ clusters because of how rapidly the number of microstates scales with N. We can pare down the number of relevant configurations in the case of attractive particles, considering only the configurations wherein particles are interacting. Taking patchy particle 2 from Figure 1 as a concrete example, we enumerate all of the clusters that can be created with two to four copies of the particle. There are 30 different $N = 2$ clusters, 736 different $N = 3$ clusters, and 25234 clusters when $N = 4$. Given any particular $N = 4$ cluster, its potential energy might range anywhere from $U = -68\varepsilon$ to $U = 26\varepsilon$, and at any of these energy levels there may be as many as 3000 distinct clusters with that energy. In this case, because we can calculate each cluster's degeneracy and energy exactly, we can calculate the partition function

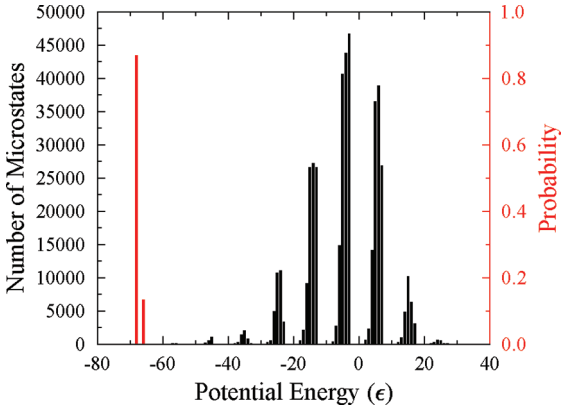


Figure 2. Frequencies (black) and relative probabilities (red) of $N = 4$ clusters of patchy particle 2 at $k_BT/\varepsilon = 1.0$ as a function of potential energy.

$$\mathcal{Z} = \sum_i \Omega_i \exp(-U_i/k_BT) \quad (1)$$

exactly, where Ω_i and U_i are the ith cluster's degeneracy and energy, respectively.[34,35] In Figure 2 we show the probability of observing a given cluster (red) and the number of configurations with a particular energy (black). Note that only two energy levels contribute non-negligibly to the partition function at this temperature. This fact may be striking to the uninitiated: less than a thousandth of a percent of the possible configurations for this system matter at this temperature.

To perform BUBBA at a given temperature, we rely on the fact that a small fraction of possible microstates comprises the majority of the partition function. This assumption breaks down at high temperatures where all microstates contribute similarly to the partition function, but is valid for the condensed soft-matter systems of interest for self-assembly. We also assume that the microstates that contribute to $\mathcal{Z}(N = n,V,T)$ must be generated by pairings of configurations from $\mathcal{Z}(N < n,V,T)$, and is independent of V (i.e., the system is relatively dilute). This allows rapid generation of partition functions hierarchically because the number of configurations that must be checked scales as a polynomial in N rather than to the power of N. In practice, we utilize a "cluster library" to keep track of each cluster generated by BUBBA. Each entry in the cluster library specifies the number, potential energy, relative degeneracy, particle positions, and combinations of smaller clusters that can create a particular cluster. In this framework, using BUBBA to calculate $\mathcal{Z}(N,V,T)$ consists simply of pairing clusters in the library, keeping only the unique microstates, and discarding the negligible microstates.

Cluster Pairing. Given a discretization of configuration space for the clusters of interest, BUBBA combines pairs of clusters to generate larger clusters. When generating size N clusters there are $\lfloor N/2 \rfloor$ combinations of cluster sizes that must be paired. For example, only $N = 1$ clusters need be combined to generate $N = 2$ clusters, but for $N = 4$ clusters we must pair $N = 1$ with

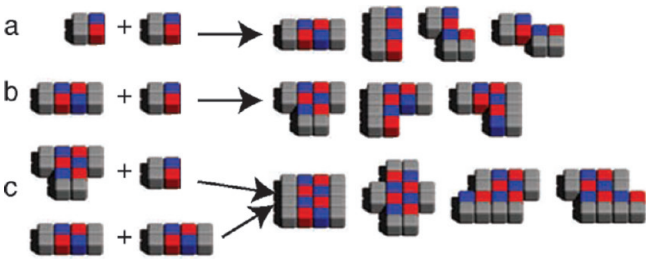


Figure 3. Examples of cluster pairing for $N = 2$ (a) $N = 3$ (b) and $N = 4$ (c). Note that for each pairing, not all possible resulting clusters are included.

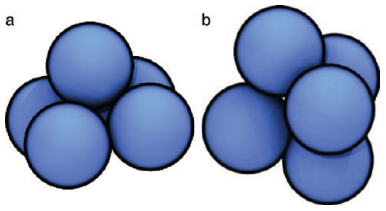


Figure 4. (a) Example of a cluster of five spheres in a pyramid structure. (b) Bipyramid structure made from the same number of spheres as in cluster a. The partition function generated by BUBBA depends upon the chosen discretization of configuration space. Different discretizations would differ in whether clusters a and b are distinguishable.

$N = 3$ clusters as well as $N = 2$ with $N = 2$ clusters (Figure 3). Each time a new candidate cluster is created from a cluster pairing, we check the cluster library to see whether it has already been generated. If the cluster is new, it is added to the cluster library. If it already exists in the library, we update the existing cluster's relative degeneracy as well as the list of cluster combinations that can generate it.

Choosing an appropriate discretization of configuration space for a particular system is crucial and it has important implications for BUBBA's performance. The chosen discretization of search space should depend on the metric by which clusters are distinguished. Figure 4a,b shows two examples of clusters made from five copies of a sphere with a continuous chain of states linking the two structures. Are they the same, or are they different? If they are different, which states count toward the degeneracy of one cluster but not the other? It is conceivable that the statistical mechanics of one system might not need to distinguish between these configurations, but that another might rely upon their difference. Whichever specific metric is chosen, be it Fourier descriptors,[36] sum of squared errors,[37] contact network analysis,[38] or some combination of methods, BUBBA requires that, for any pair of clusters, they can be classified as being the same or different, and that a finite set of allowed pairings of the two clusters can be generated.

Distinguishing Clusters. To distinguish between configurations, we define a graph for each cluster and compare traversals of each graph.[39] Here, each cell in a tetromino defines a node, and edges (connections) between nodes are defined by cell adjacencies (Figure 5). If any one traversal of one graph matches a particular traversal of the second graph, the two configurations are considered identical; otherwise they are different. A traversal is a tree of nodes where each node is visited exactly once. When comparing two traversals, they are equal if the node types and connectivities are

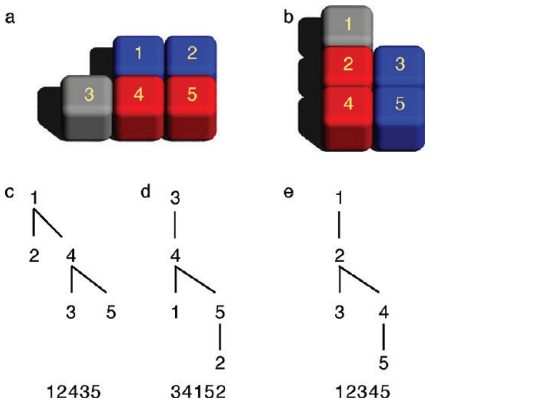


Figure 5. (a) An example cluster, with numbered node labels. (b) Reference cluster with numbered node labels. (c) One possible graph traversal of cluster a that does not match the particular graph traversal of cluster b. The sequence of nodes visited during a traversal are shown below the graphs and are generated by first visiting the root (top) node and following branches depth-first, left-to-right, not counting backtraces, until each node has been visited. (d) Graph traversal of cluster a that matches the particular graph traversal of cluster b. (e) Particular graph traversal of cluster b that is used as the reference structure. When traversing a graph, the number of nearest neighbors for the cell and the cell type are compared at each node.

identical for both traversals. Here, two nodes are identical if they have the same type (red, blue, or gray), and if they have the same number of connections. The two optimizations that reduce the number of graph traversals that we implement compare cluster energies, length, and width before beginning a graph traversal.

Discarding Negligible Clusters. Once a partition function has been generated, we discard all the smallest-weighted clusters whose summed weights are less than a chosen cutoff c. Typically, we choose c in the range $0 \leq c \leq 0.01$ because c is the error in the $N = 2$ partition function, and this error will compound roughly as c^N. Next, we sort the clusters by their normalized Boltzmann weights

$$w_i = \Omega_i \exp(-U_i/k_B T)/\mathcal{Z} \quad (2)$$

where $\mathcal{Z}$ is the partition function from eq 1.[10] By summing the Boltzmann weights in order, beginning with the largest, we find the clusters for which their sum of weights is less than c, which we then discard. To optimize the calculation of cluster probabilities, we sort the clusters by $r_i = -U_i + k_B T \ln(\Omega_i)$. Using r_i we calculate w_i with the formula $w_i = \exp(r_i - r_0)/\Sigma_j \exp(r_j - r_0)$, where r_0 is for the cluster with the largest Boltzmann weight.

Calculating the degeneracies Ω_i for lattice model clusters can be performed exactly and must be approximated for clusters with continuous degrees of freedom. For a lattice cluster that can be made from p pairings where each pairing is defined by clusters a and b,

$$\Omega_i = \sum_p \Omega_a \Omega_b n_p \quad (3)$$

where n_p is the number of distinct ways of combining a and b to make cluster i.

For systems with continuous degrees of freedom, it is not straightforward to sum over the same set of cluster combinations because of constraints placed on a cluster's degrees of freedom when the two clusters are combined.

The relative degeneracies

$$\Omega_i = q_{i,\mathrm{rot}} q_{i,\mathrm{vib}} \tag{4}$$

for such off-lattice clusters should be calculated with the rotational ($q_{i,\mathrm{rot}}$) and vibrational ($q_{i,\mathrm{vib}}$) partition functions.[10,40,41] The rotational partition function for a general rigid body

$$q_{\mathrm{rot}} = \frac{\pi^{1/2}}{\sigma}\left(\frac{8\pi^2 I_x k_B T}{h^2}\right)^{1/2}\left(\frac{8\pi^2 I_y k_B T}{h^2}\right)^{1/2}\left(\frac{8\pi^2 I_z k_B T}{h^2}\right)^{1/2} \tag{5}$$

where I_x, I_y, and I_z are the diagonal elements of the rigid body's diagonalized inertial tensor $\mathscr{I}$, σ is the body's symmetry number, and h is Planck's constant.[10] The calculation of $\mathscr{I}$ can be performed using

$$\mathscr{I} = \sum_k \mathscr{I}_k + m_k[(\boldsymbol{r_k}\cdot\boldsymbol{r_k})E_3 - \boldsymbol{r_k}\otimes\boldsymbol{r_k}] \tag{6}$$

where the sum is over all particles in the cluster, $\mathscr{I}_k$ is the inertial tensor of the kth particle about its center of mass, m_k is the kth particle's mass, $\boldsymbol{r_k}$ is a displacement vector from the cluster's center of mass to the kth particle's center of mass, E_3 is the 3×3 identity matrix, and $\otimes$ is the dyadic product. For clusters of the same size at a particular temperature, eq 5 simplifies to

$$q_{\mathrm{rot}} = \frac{(I_x I_y I_z)^{1/2}}{\sigma} \tag{7}$$

because only the relative magnitudes of the rotational partition functions for clusters of the same size matter for their probabilities. To determine the symmetry number σ, we count all linearly independent combinations of rotation operations that map a cluster onto itself. For a cluster defined by the types and positions of its particles, we try all rotation operations that map one particle i onto all other particles of the same type. For each of these, we attempt a second rotation about an axis passing through i until a second particle j is aligned with a reference particle of the same type. After this alignment we calculate the sum of squared errors

$$\delta = \sum_k |\vec{r}_k - \vec{r}_0|^2 \tag{8}$$

where $\vec{r}_k$ is the position of the kth particle and $\vec{r}_0$ is the position of the reference particle of the same type closest to $\vec{r}_k$. If this trial orientation has not been previously generated and if $\delta < 0.001$, we increment σ.

Calculations of the vibrational partition function can also be performed numerically for off-lattice clusters using Einstein oscillators or any analogous method for determining the free volumes accessible to each component of a cluster.[10,40] Generally,

$$q_{\mathrm{vib}} = \prod_k V_k \tag{9}$$

for clusters with no collective vibrational modes, where V_k is the volume accessible to the kth particle in the cluster. For example, the vibrational degrees of freedom matter substantially for clusters in Figure 4 if they are made up of five hard spheres that can move on the surface of a hidden sphere. In this case, an Einstein oscillator approximation of q_{vib} would overpredict the entropic contribution from the sphere at the apex of the pyramid in Figure 4, but may under-predict the contributions from the "base" spheres.

Table 1. Minutes Required to Generate Partition Functions for All 25 Building Blocks in Figure 1 up to $N = 10$ Using BUBBA

	$c = 0.01$	$c = 0.001$	$c = 0.0001$
$k_BT/\varepsilon = 0.1$	3	3	4
$k_BT/\varepsilon = 1.0$	4	35	847
$k_BT/\varepsilon = 3.0$	4	513	1320

Table 2. Average Partition Function Error (eq 10) for All Patchy Particles in Figure 1, Excluding Particle 6

	$e(0.01)$	$e(0.001)$	$e(0.0001)$
$k_BT/\varepsilon = 0.1$	0.000	0.000	0.000
$k_BT/\varepsilon = 1.0$	0.007	0.007	0.004
$k_BT/\varepsilon = 3.0$	0.025	0.016	0.004

Here, we integrate the accessible volume to each sphere as a fraction of the total center sphere surface area with Monte Carlo integration. This approach generalizes to arbitrary cluster types and interaction types between the subunits of the cluster.

RESULTS/DISCUSSION

In using BUBBA to generate partition functions of a given size N, the cutoff value c can be chosen to strike a balance between efficiency and accuracy. Small values of c ensure the partition functions built at each N are more complete, at the cost of including more cluster configurations. Of course, as k_BT/ε increases, more clusters contribute to the partition function, so choosing c too large will cause BUBBA to miss important configurations. Table 1 summarizes the relationship between BUBBA runtime, c, and k_BT/ε for the 25 model particles in Figure 1. Note that partition functions up to $N = 4$ can be easily generated when $c = 0$, and an extrapolation of the resulting trend suggests that generating an $N = 10$ partition function for $c = 0$ would require $O(10^8)$ years with the current implementation.

To quantify the accuracy of the partition functions generated by BUBBA we compare the $N = 4$ clusters generated at $c > 0$ with the clusters generated when $c = 0$. The error

$$e(c) = \sum_i (w_{c=0,i} - w_{c,i})^2 \tag{10}$$

where $w_{c,i}$ is the Boltzmann weight of the ith cluster generated by BUBBA with cutoff c. We find that for any given building block, the required c to generate accurate partition functions will depend strongly on temperature. For example, at $k_BT/\varepsilon = 0.1$, we find $e(0.00001) = 0$ for all 25 particles in Figure 1, indicating no clusters were discarded at this value of c. For $0.0001 \le c \le 0.01$ and $k_BT/\varepsilon = 0.1$, $e(c) = 0$ for all particles except for particle 6, for which $e(c) = 2.0$ because the lowest energy cluster is missed. Considering only the patchy particles in Figure 1, we find that $e(c)$ roughly scales with c (Table 2) and k_BT/ε. We exclude the nonpatchy particles from this analysis because the characteristic temperature ranges differ substantially from those for the patchy particles, making comparisons less useful.

BUBBA's efficiency at low temperatures allows for clusters larger than $N = 10$ to be generated easily. As a case study, we consider the partition functions for patchy particle 6 at $k_BT/\varepsilon = 0.1$, $c = 0.00001$, and $N \le 50$. Figure 6 displays the elapsed runtime for these conditions on a 2.6 GHz Intel processor.

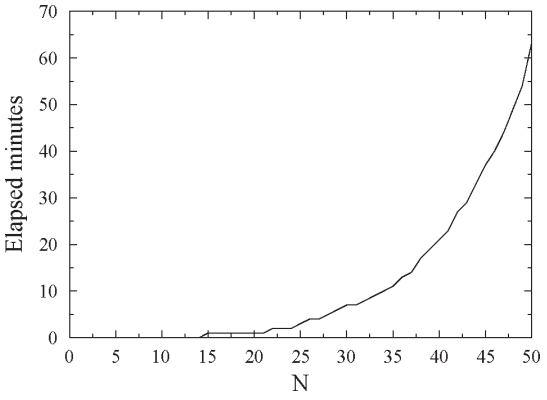


Figure 6. Minutes of runtime to generate the Nth partition function for patchy particle 6 at $k_BT/\varepsilon = 0.1$ and $c = 0.00001$ using BUBBA.

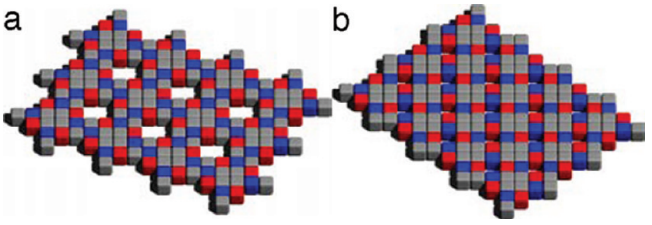


Figure 7. (a) Most likely $N = 48$ configuration at $k_BT/\varepsilon = 0.1$ for patchy particle 6. There are 11,361,667,066,181,451,776 ways to generate this cluster from combinations of distinguishable patchy particles. (b) 20th most likely $N = 48$ configuration at $k_BT/\varepsilon = 0.1$ for patchy particle 6. This cluster is less likely than cluster a, despite their identical energies, because there are only 1,972,576,636,788,277,248 ways to generate it.

For $N > 25$ this data fits a weak exponential with the number of minutes $t = 0.2 \exp(0.12\ N)$. At large cluster sizes it is apparent that periodic tilings of the patchy particle are the most thermodynamically stable arrangements (Figure 7). At these conditions, the more porous tiling in Figure 7a is only 5.8 times more likely than the tiling in Figure 7b, and this difference in entirely entropic in origin. The two tilings have the same number of building blocks, and identical energies; however, there are exactly 9,389,090,429,393,174,528 more ways to make the more porous tiling than the 1,972,576,636,788,277,248 ways of generating the tiling in Figure 7b, as calculated by BUBBA with $c = 0.00001$. The existence of these two competing tilings has negative implications for this building block's propensity for self-assembly of either pattern at this temperature. Systems of building blocks for which two or more structurally distinct tilings are commensurate in probability will fail to assemble any tiling or pattern robustly because of the enormous energetic penalty to rearrange one competing structure into the other. We confirm that patchy particle 6 does not self-assemble porous arrays robustly via Monte Carlo simulations, and expand upon the identification of robust assembly candidates in a forthcoming manuscript.[42]

We determine thermodynamically stable motifs for all of the 25 particles in Figure 1 at $k_BT/\varepsilon = 0.1$ in the same way as described above for patchy particle 6. Tilings of the unit cells identified in the most probable $40 \leq N \leq 50$ clusters are presented in Figure 8. Although the family of model particles considered here was chosen primarily to provide a proof-of-concept demonstration of our approach, it is easy to envision possible applications of some of the motifs in Figure 8a—l. For example, some of these

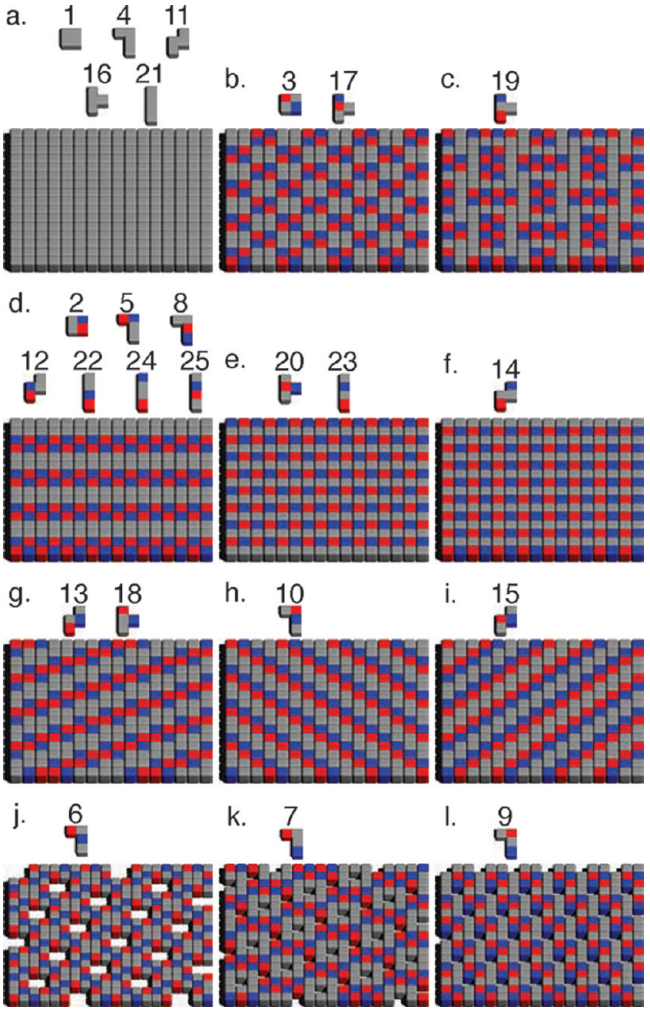


Figure 8. Energy minimizing motifs stabilized at $k_BT/\varepsilon = 0.1$ for the 25 particles in Figure 1, from patterns in $N = 50$ clusters generated with BUBBA.

patterns (e.g., Figure 8c—i) could be used as nanoscale circuit elements.[43,44] If the red and blue cells are made of an insulating material while the gray are conductive, these building blocks can be used to assemble arrays of wires separated by insulating barriers. Patchy particle 19 could also be used to generate a structure with potentially interesting electronic applications: its continuous gray domains (Figure 8c) could be used in a service that requires a redundant conducting path through an otherwise insulating matrix. The porous arrays of Figure 8j—l have potential applications in nanoscale filtration[45] and fuel cells.

CONCLUSIONS

We have demonstrated that the generation of partition functions hierarchically to arbitrary precision is efficient and straightforward, and that it can aid in assessing self-assembly propensity. In the cases where small numbers of distinguishable configurations comprise a majority of a partition function's weight, as is the case for systems at low temperatures and for many anisotropic building blocks with disparate interactions, BUBBA is a particularly effective method for generating partition functions that have been heretofore inaccessible. For the first time, this allows the question of

"What structures are thermodynamically favored for this building block at any temperature?" to be answered independent of assembly kinetics. Finally, for the family of model patchy particles studied here we discover a diverse set of thermodynamically stable structures that may have practical applications at the nanoscale.

The current implementation of BUBBA is effective and demonstrative, but there are many opportunities for improvement and application. Many elements of the algorithms used stand to benefit substantially from parallelization of inner loops. For example, there is no reason the shape-matching of one cluster against thousands of clusters in the cluster library must execute serially, and a GPU implementation of this aspect alone would enable partition functions larger by an order of magnitude to be generated. BUBBA also has promise as an efficient screening tool that would allow for the identification of the parts of a partition function that are consistent or inconsistent with the assembly of target structures. Using BUBBA as a screening tool, implementing kernels on GPU architectures, and extending to other ensembles are the subjects of ongoing work.

AUTHOR INFORMATION

Corresponding Author
*E-mail: sglotzer@umich.edu.

ACKNOWLEDGMENT

This material is based upon work supported by the DOD/DDRE under Award No. N00244-09-1-0062, the National Science Foundation Award No. CHE 0624807, and the James S. McDonnell Foundation 21st Century Science Research Award/Studying Complex Systems, grant no. 220020139. This research was made possible with Government support under and awarded by DoD, Air Force Office of Scientific Research, National Defense Science and Engineering Graduate (NDSEG) Fellowship, 32 CFR 168a (E.J.). Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the DOD/DDRE. E.J. also thanks Aaron Santos and Carolyn Phillips for useful discussions.

REFERENCES

(1) Whitesides, G.; Grzybowski, B. *Science* **2002**, *295*, 2418–2421.
(2) Nie, Z.; Petukhova, A.; Kumacheva, E. *Nat. Nanotechnol.* **2010**, *5*, 15–25.
(3) Bates, F. S.; Fredrickson, G. H. *Annu. Rev. Phys. Chem.* **1990**, *41*, 525–557.
(4) Park, H.-S.; Agarwal, A.; Kotov, N. A.; Lavrentovich, O. D. *Langmuir* **2008**, *24*, 13833–13837.
(5) Rothemund, P. W. K. *Nature* **2006**, *440*, 297–302.
(6) Tirrell, M. *AIChE J.* **2005**, *51*, 2386–2390.
(7) Glotzer, S. C.; Solomon, M. J. *Nat. Mater.* **2007**, *6*, 557–562.
(8) Zhang; Glotzer, S. C. *Nano Lett.* **2004**, *4*, 1407–1413.
(9) Wales, D. J.; Bogdan, T. V. *J. Phys. Chem. B* **2006**, *110*, 20765–20776.
(10) McGinty, D. J. *J. Chem. Phys.* **1971**, *55*, 580–588.
(11) McQuarrie, D. A. *Statistical Mechanics*; University Science Books: Sausalito, CA, 2000.
(12) Baxter, R. J. *Phys. Rev. Lett.* **1984**, *53*, 1795–1798.
(13) Mertens, S. *Theor. Comput. Sci.* **2001**, *265*, 79–108.
(14) Frenkel, D.; Smit, B. *Understanding Molecular Simulations: From Algorithms to Applications*; Elsevier, 2002.
(15) Georges, A.; Kotliar, G.; Krauth, W.; Rozenberg, M. J. *Rev. Mod. Phys.* **1996**, *68*, 13.
(16) Poole, P. H.; Sciortino, F.; Essmann, U.; Stanley, H. E. *Nature* **1992**, *360*, 324–328.
(17) Srinivas, G.; Discher, D. E.; Klein, M. L. *Nat. Mater.* **2004**, *3*, 638–644.
(18) Zhang, Z.; Horsch, M. A.; Lamm, M. H.; Glotzer, S. C. *Nano Lett.* **2003**, *3*, 1341–1346.
(19) Nguyen, T. D.; Zhang, Z.; Glotzer, S. C. *J. Chem. Phys.* **2008**, *129*, 244903.
(20) Rapaport, D. *Phys. Rev. E.* **2004**, *70*, 051905.
(21) Nguyen, H. D.; Reddy, V. S.; Brooks, C. L. *Nano Lett.* **2007**, *7*, 338–344.
(22) Haji-Akbari, A.; Engel, M.; Keys, A. S.; Zheng, X.; Petschek, R. G.; Palffy-Muhoray, P.; Glotzer, S. C. *Nature* **2009**, *462*, 773–777.
(23) Barnes, B. C.; Siderius, D. W.; Gelb, L. D. *Langmuir* **2009**, *25*, 6702–16.
(24) Benjamin, K. M.; Schultz, A. J.; Kofke, D. A. *J. Phys. Chem. C* **2007**, *111*, 16021–16027.
(25) Zhang, Z.; Tang, Z.; Kotov, N. A.; Glotzer, S. C. *Nano Lett.* **2007**, *7*, 1670–5.
(26) Bryngelson, J. D.; Wolynes, P. G. *Proc. Natl. Acad. Sci. U.S.A.* **1987**, *84*, 7524–7528.
(27) Karplus, M.; Sali, A. *Curr. Opin. Struct. Biol.* **1995**, *5*, 58–73.
(28) Jankowski, E.; Glotzer, S. C. *J. Chem. Phys.* **2009**, *131*, 104104.
(29) Troisi, A.; Wong, V.; Ratner, M. A. *Proc. Natl. Acad. Sci. U.S.A.* **2005**, *102*, 255–60.
(30) Cicoira, F.; Rosei, F. *Surf. Sci.* **2006**, *600*, 1–5.
(31) Schneider, T. M.; Mandre, S.; Brenner, M. P. *Phys. Rev. Lett.* **2011**, *106*, 094503.
(32) Sung, K. E.; Vanapalli, S. A.; Mukhija, D.; McKay, H. A.; Millunchick, J. M.; Burns, M. A.; Solomon, M. J. *J. Am. Chem. Soc.* **2008**, *130*, 1335–40.
(33) Li, F.; Josephson, D. P.; Stein, A. *Angew. Chem., Int. Ed. Engl.* **2011**, *50*, 360–88.
(34) Stillinger, F. H.; Weber, T. A. *Science* **1984**, *225*, 983–989.
(35) Strodel, B.; Wales, D. J. *Chem. Phys. Lett.* **2008**, *466*, 105–115.
(36) Zahn, C. T.; Roskies, R. Z. *IEEE Trans. Comput.* **1972**, *C-21*, 269–281.
(37) Besl, P.; McKay, N. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **1992**, *14*, 239–256.
(38) Keys, A. S.; Iacovella, C. R.; Glotzer, S. C. *Annu. Rev. Condens. Matter Phys.* **2011**, *2*, 263–285.
(39) Christofides, N. *Omega* **1973**, *1*, 719–732.
(40) Foffi, G.; Sciortino, F. *Phys. Rev. E* **2006**, *74*, 050401.
(41) Meng, G.; Arkus, N.; Brenner, M. P.; Manoharan, V. N. *Science* **2010**, *327*, 560–3.
(42) Jankowski, E.; Glotzer, S. C. To be submitted for publication, 2011.
(43) Gudiksen, M. S.; Lauhon, L. J.; Wang, J.; Smith, D. C.; Lieber, C. M. *Nature* **2002**, *415*, 617–620.
(44) McAlpine, M. C.; Friedman, R. S.; Jin, S.; Lin, K.-h.; Wang, W. U.; Lieber, C. M. *Nano Lett.* **2003**, *3*, 1531–1535.
(45) Gopal, R.; Kaur, S.; Ma, Z.; Chan, C.; Ramakrishna, S.; Matsuura, T. *J. Membr. Sci.* **2006**, *281*, 581–586.