A comparison of new methods for generating energy-minimizing configurations of patchy particles

Eric Jankowski and Sharon C. Glotzer

Citation: J. Chem. Phys. **131**, 104104 (2009); doi: 10.1063/1.3223834
View online: http://dx.doi.org/10.1063/1.3223834
View Table of Contents: http://jcp.aip.org/resource/1/JCPSA6/v131/i10
Published by the American Institute of Physics.

A comparison of new methods for generating energy-minimizing configurations of patchy particles

Eric Jankowski[1,a)] and Sharon C. Glotzer[1,2,b)]
[1]*Department of Chemical Engineering, University of Michigan, Ann Arbor, Michigan 48109, USA*
[2]*Department of Materials Science and Engineering, University of Michigan, Ann Arbor, Michigan 48109, USA*

(Received 27 March 2009; accepted 17 August 2009; published online 9 September 2009)

Increasingly complex particles are pushing the limits of traditional simulation techniques used to study self-assembly. In this work, we test the use of a learning-augmented Monte Carlo method for predicting low energy configurations of patchy particles shaped like "Tetris®" pieces. We extend this method to compare it against Monte Carlo simulations with cluster moves and introduce a new algorithm—bottom-up building block assembly—for quickly generating ordered configurations of particles with a hierarchy of interaction energies. © *2009 American Institute of Physics*. [doi:10.1063/1.3223834]

I. INTRODUCTION

Patchy particles, including those possessing intricate geometries and hierarchies of interaction energies, hold promise as the building blocks for next-generation materials and devices.[1] However, the complicated interactions between particles make the prediction of their energy-minimizing ordered arrangements challenging when using essentially all particle-based simulation methods rooted in statistical thermodynamics. This is particularly true of stochastic methods such as Monte Carlo (MC). Despite optimizations[2] that obey detailed balance such as cluster moves,[3,4] hybrid MC,[5] and configurational bias methods,[6,7] MC simulations can become intractably long when used to predict energy -minimizing configurations of patchy particles with disparate interaction energies because trial move acceptance probabilities become so low that the simulation becomes stuck in a suboptimally organized, kinetically arrested state. Consequently, there is renewed interest in the development of simulation methods for strongly interacting, complex molecular and particle-based systems.[4,8]

A new method proposed by Troisi *et al.* is an important step in this direction. The authors introduce an adaptive, learning-augmented MC (LAMC) approach intended to prevent the formation of suboptimal local arrangements of model complex building blocks shaped like "Tetris®" pieces (Fig. 1).[9] On a two-dimensional lattice, these building blocks are examples of lattice animals.[10] They record the energies of the best (i.e., lowest energy) clusters up to a maximum size, updating these values any time a new, lower energy cluster is found. Any cluster of particles selected for a trial move that does not have an energy consistent with the optimal value for its size is forced to split apart. Biasing the formation of low-energy clusters in this way, the simulation learns about successively better clusters as it proceeds. By moving a best-energy (i.e., lowest energy) cluster of a given size as a single unit, Troisi *et al.*'s LAMC benefits from two performance improvements over traditional MC. The first improvement is that trial moves breaking apart the best-energy clusters are never attempted until a better energy cluster is discovered. Second, the simultaneous translations and rotations of best-energy clusters help overcome the long agglomeration times that occur in MC simulations without cluster moves. As a result, their method generates very low temperature ordered structures faster than traditional MC.[9]

However, by not allowing cluster moves for suboptimal clusters, the LAMC algorithm loses some efficiency in exploring phase space. It is tempting to think that a Monte Carlo simulation that includes cluster moves for all clusters in addition to Troisi *et al.*'s learning-augmented cluster bias will find energy-minimizing configurations faster than either LAMC or cluster MC (cMC). In cMC, the random translation that would normally be applied to a single particle during a trial move is applied to a group of particles (cluster) that are defined based on their relative positions. Rotations of a cluster can also be included. Therefore, a simulation that combines cluster moves and Troisi *et al.*'s learning augmentation would be expected to overcome long agglomeration times better than LAMC and would utilize fewer trial moves that attempt to break optimal configurations than cMC. Thus, such a method could potentially be more efficient than cMC for generating low temperature states.

This work is divided into three main sections. In the first section we develop a new learning-augmented cMC (LAcMC) method following the above line of reasoning and compare it against cMC. In the second section, we are inspired by Troisi *et al.*'s bottom-up cluster bias approach to develop a new algorithm—bottom-up building block assembly (BUBBA)—for quickly generating energy-minimizing configurations of complex building blocks. In the final section we provide an analysis of each method's ability to find low energy structures and their relative efficiencies, and dis-

[a)]Electronic mail: erjank@umich.edu.
[b)]Electronic mail: sglotzer@umich.edu.

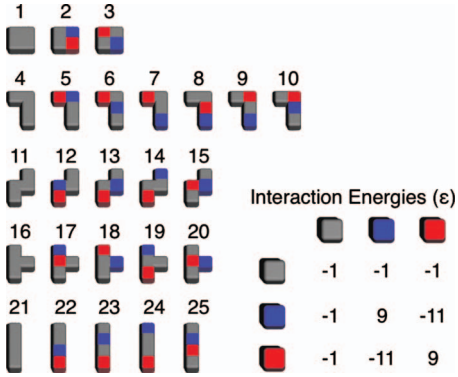

FIG. 1. The 25 primitive particles studied in this work and in Troisi *et al.* (Ref. 9). Gray represents neutral subunits, red are positive, and blue are negative.

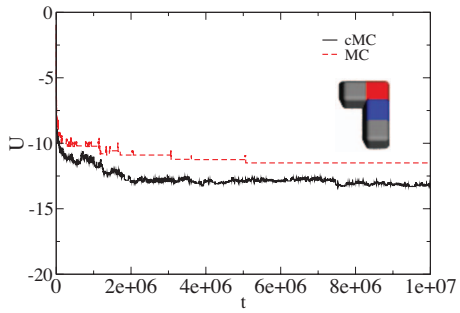

FIG. 2. Per-particle potential energy profiles for cMC and MC simulations of patchy particle 10, with 40 particles in a 32×32 lattice at $1/\beta\epsilon=1.4$.

cuss the theoretical and practical limitations of each method. In all sections, we utilize the same model system, which is described in the next section.

II. MODEL SYSTEM

To facilitate comparison with previous work, we consider systems of particles identical to those invented by Troisi *et al.*[9] Each rigid tetromino particle is made up of four subunits and is shaped like a Tetris® piece. Each subunit is defined to be positive, negative, or neutral. All nearest neighbor subunits interact with a weak van der Waals-like attraction that has a potential energy value of $U=-\epsilon$. Additionally, the interaction of nearest-neighbor charged subunits is $U=\pm10\epsilon$, depending on their respective charges. As such, the energy associated with a neutral subunit interacting with any other subunit type is $-\epsilon$, that of oppositely charged subunits is $U=-11\epsilon$, and the energy between like charges is $U=+9\epsilon$. The 25 different primitive particle types studied here and by Troisi et *et al.* are shown in Fig. 1. These building blocks can translate and rotate on a two-dimensional square lattice, but particle overlaps, inversions, deletions, substitutions, and additions are prohibited. Extensions of the methods investigated here to three-dimensional lattices, other lattice geometries, and off-lattice are possible but beyond the scope of this paper.

III. LEARNING AUGMENTATION AND CLUSTER MONTE CARLO

The cMC simulation we develop for the particles in Fig. 1 utilizes cluster moves similar to those used by Whitelam *et al.*[4] At a given time step i, particles k and l are considered part of the same cluster with probability $p_{k,l}^{i}=0.5$ if particles k and l share an edge in time step i and $p_{k,l}^{i}=0$ otherwise. To ensure the trial move for the cluster C selected in each time step maintains detailed balance, the trial move acceptance probability is

$$\mathrm{acc}(o\rightarrow n)=\min\left(1,e^{-\beta(U_n-U_o)}\frac{\prod_{l\notin C}^{k\in C}(1-p_{k,l}^{n})}{\prod_{l\notin C}^{k\in C}(1-p_{k,l}^{o})}\right),\qquad(1)$$

where β is the inverse temperature $1/k_BT$, k_B is Boltzmann's constant, U_n is the potential energy of the new configuration, and U_o is the potential energy of the old configuration. A detailed derivation of Eq. (1) is included in Appendix. The choice of bond probability $p_{k,l}$ for a simulation is arbitrary, but can have a significant impact on equilibration times.[4,11,12] Bond probabilities based on interaction energies are employed in rejection-free cluster moves for spin glasses[13] and can be generalized to fluid systems,[12] but are prone to unphysical kinetic traps when particle interactions are above a few k_BT.[4] We choose a constant bond probability specifically because it avoids the unphysical kinetic traps associated with rejection-free cluster moves in the same way as Whitelam and Geissler's cleaving algorithm[4] and for its computational efficiency. Empirically, we find $p_{k,l}=0.5$ strikes a balance between the growth of clusters (high $p_{k,l}$) and the rate at which their trial moves are accepted (low $p_{k,l}$). The details of our depth-first cluster generation algorithm are included in Appendix. We find MC simulations that include this implementation of cluster moves to be more efficient than those without, as expected. Figure 2 shows the trajectories of a cMC and standard MC simulation with identical parameters and initial conditions. For all 25 building blocks, the simulations with cluster move sample lower energy configurations, again as expected.

Our LAcMC code is identical to cMC except we do not allow the best clusters in time step t to split apart in time step $t+1$. In LAcMC, we define clusters every time step as in cMC and compare the potential energy of each cluster with the optimal energy recorded for clusters of that size. If a cluster's potential energy is equal to the optimum for its size, we set $p_{k,l}^{t+1}=1$ for all pairs of particles in the cluster. If the potential energy of a cluster is lower than the recorded optimum, we update the new optimum value. With this algorithm, our LAcMC model is identical to Troisi *et al.*'s LAMC method except that here clusters with less-than-optimal potential energies are allowed to move as whole units. Thus, LAcMC combines the energy-minimizing cluster bias of LAMC and the agglomeration-optimizing cluster moves of cMC.

For each of the 25 patchy particles in Fig. 1, we perform 500 cMC simulations and 500 LAcMC simulations of 5×10^5 time steps, each with a unique random number generator seed. The simulations are performed on a 32×32 lattice with periodic boundary conditions initialized randomly with $N=40$ identical particles. The temperature is set to $1/\beta\epsilon=0.06$ for the patchy particles 1, 4, 11, 16, and 21. For

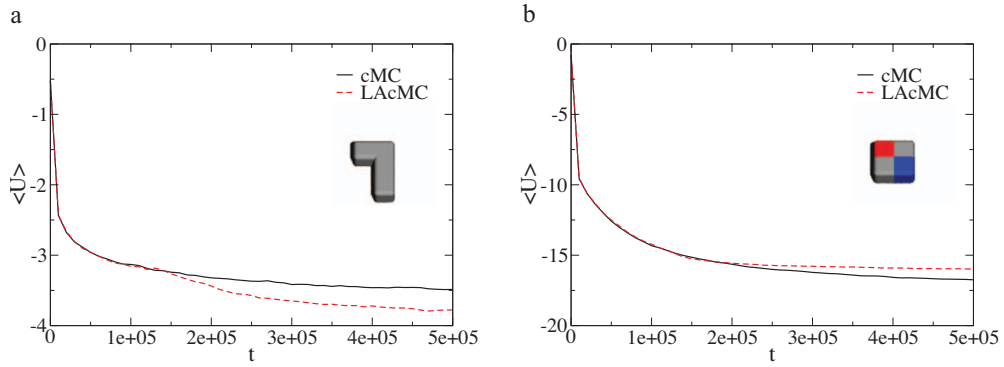

FIG. 3. Per-particle potential energy histories of (a) patchy particle 3 and (b) patchy particle 4 simulations for cMC and LAcMC. Standard error bars are on the order of the line thickness.

patchy particles 6, 10, 15, 17, and 20, the temperature is set to $1/\beta\varepsilon=1.4$. The reduced temperature for all other patchy particles is set to $1/\beta\varepsilon=2.2$. These temperatures are chosen to be low enough to facilitate agglomeration, but high enough to prevent the system from becoming trapped in an arrested state and were determined by trial and error.

To compare the rates at which cMC and LAcMC converge upon potential energy minima, we average the per-particle potential energy histories of the 500 trials for each method. Figure 3(a) presents the histories for patchy particle 4, where LAcMC finds lower energy configurations on average. For patchy particle 3, however, the opposite result is obtained with cMC outperforming LAcMC [Fig. 3(b)]. Table I tabulates whether LAcMC performs on average better than (−), worse than (+), or the same as (○) cMC for each patchy particle. In general, we find no obvious correlation between particle shape or patterning and whether or not LAcMC will tend to find better-optimized configurations than cMC without learning.

Because LAcMC is an extension of cMC, it is natural to ask how the addition of an adaptive element alters the ensemble sampled. Figure 4(a) presents a comparison of the cluster size distributions for patchy particle 23 when simulated with learning-augmented and non-LAcMC. This distribution is representative of those found for all 25 patchy particles. The shift in LAcMC toward larger clusters demonstrates that it is not sampling the same distribution of microstates as cMC, since the cluster size distributions must be a property of the state point. This discrepancy is expected, since the algorithm is designed to bias toward low-energy (that is, large) clusters, but highlights the fact that LAcMC cannot be used to sample a thermodynamic ensemble away from zero temperature.

Furthermore, any algorithm similar to LAMC or LAcMC in which the system "learns" can never be tuned to sample a thermodynamic ensemble. The non-Markovian nature of this type of algorithm, where the transition made from time step t to $t+1$ depends not only on the state at t, but on a history of states, precludes the condition of balance necessary for ergodic sampling.[14] There is no variant of Eq. (1) that can rectify the state sampling differences caused by learning augmentation. However, the fact that LAcMC cannot be used at finite temperature is an acceptable shortcoming so long as it outperforms standard methods in the low temperature energy-minimizing case, where standard methods typically fail.

In Table II we compare the lowest-energy configuration found by LAcMC with those found by cMC, averaged over 500 runs for each method. LAcMC does not reproducibly find lower energy structures than cMC, nor does it find them in fewer timesteps (Table III). Tables I and II are in close agreement, as we would expect: if the average configurations generated by LAcMC for a given patchy particle have lower energies than those generated by cMC, then the lowest-energy configurations generated by LAcMC are also lower than the best structures generated by cMC. However, the averages and standard errors presented in Table II fail to paint the full picture of how LAcMC changes the distribution of configurations sampled. Figure 4(b) demonstrates that LAcMC samples a much narrower distribution of energies with a lower average for patchy particle 21. In this case, the fact that learning augmentation improves the sampling of lower energy configurations is lost in the averages of Table II, but clearly appears in the side-by-side comparison of the energy distributions sampled by each method.

For some particles (e.g., patchy particle 9), the distribution of best-energy configurations is shifted lower than that for cMC, indicating that the cluster biasing facilitates finding lower energy structures [Fig. 4(c)]. For other patchy particles (e.g., patchy particle 3), the distribution of best-energy configurations found by LAcMC is worse than that found by cMC [Fig. 4(d)]. However, regardless of learning augmentation's ability to find lower energy configurations, the cluster size distribution is shifted as in Fig. 4(a) for all patchy particles. No general improvement is found for LAcMC over cMC.

While LAcMC and cMC take statistically similar amounts of time steps to find best-energy configurations (Table III), LAcMC offers a small amount of real-time per-

TABLE I. Impact of learning augmentation on average final potential energy after 500 000 time steps of cMC. A "+" indicates LAcMC finds higher (worse) U configurations than cMC, "−" indicates lower (better), and "○" indicates no difference.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
-	-	+	-	-	+	+	+	-	o	+	+	-	+	+	-	-	-	o	-	-	-	-	+	-

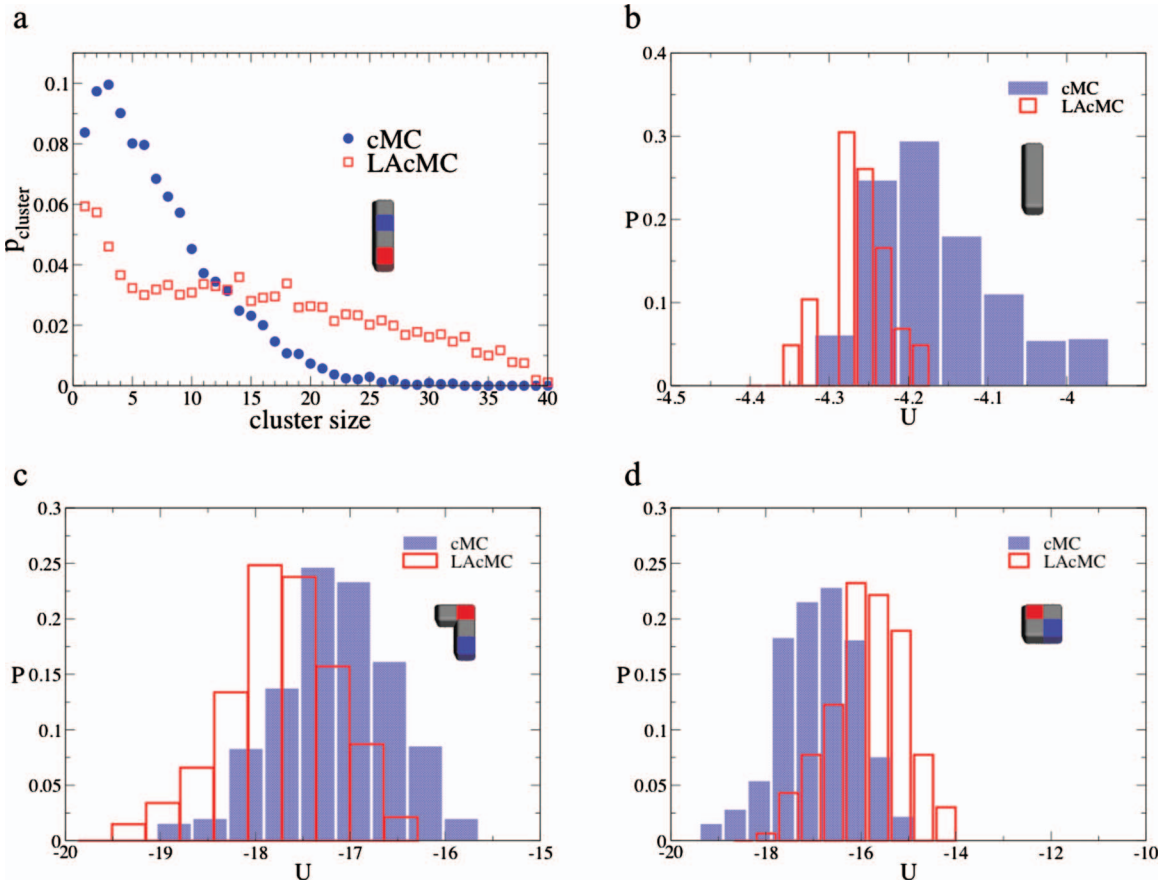

FIG. 4. Learning augmentation changes the distribution of cluster sizes during a simulation, as well as the distribution of energies sampled. (a) Cluster size distribution for patchy particle 23 from simulations with $N=40$, $1/\beta\varepsilon=0.1$ on a 32×32 lattice. (b) Distributions of the best energy configuration found for patchy particle 21. (c) Distributions of the best energy configuration found for patchy particle 9. (d) Distributions of the best energy configuration found for patchy particle 3.

formance improvement. Because energy-minimizing clusters in time step t are maintained in time step $t+1$, the number of clusters that need to be generated in later time steps are substantially decreased. Because of this, a cMC simulation of 5×10^5 time steps takes on average 52.4 seconds on a 2.8 Ghz iMac, and an identical LAcMC simulation takes an average of 39.3 s.

IV. BOTTOM-UP BUILDING BLOCK ASSEMBLY

In both the LAMC method developed by Troisi *et al.* and our LAcMC method, the expectation is that energy-minimizing configurations of particles can be found more effectively by biasing the formation of low-energy clusters. However, we observe that adding a learning-augmented cluster bias component does not offer a significant performance increase over traditional methods. This does not imply that learning-augmented methods cannot be used or that they should be avoided. In BUBBA, we take this idea of learning and cluster biasing one step further and build energy-minimizing clusters from pairs of smaller clusters from the bottom up. To generate an energy-minimizing cluster of size N from smaller best-energy clusters, BUBBA employs Algorithm 1. This algorithm generates the lowest energy clusters of size N by checking all configurations for every pair of clusters whose sizes sum to N. Here, we consider a simplification of BUBBA where only pairs of lowest energy clusters are used to build larger clusters. This implementation of BUBBA is efficient but will miss energy-minimizing clusters made from suboptimal clusters, and we return to this point in Sec. V.

Algorithm 1 Generate clusters of size N

Require: generation of clusters from size 1 through $N-1$

```
for all s such that N/2 ≤ s ≤ N−1 do
  for all clusters y ∈ clustersOfSize[s] do
    for all clusters m ∈ clustersOfSize[N−s] do
      B ← 0
      calculate perimeter of y
      for all cells cy in perimeter of y do
        for all cells cm in cluster m do
          move m such that cm=cy
          for all orientations do
            E ← energy(m,y)
            if E<B then
              B ← E
              replace bestClustersOfSize[N] with cluster(m,y)
            if E=B and cluster(m,y) ∉ bestClustersOfSize[N] then
              add cluster(m,y) to bestClustersOfSize[N]
return bestClustersOfSize[N]
```

Table IV presents the per-particle potential energies of the size $N=40$ clusters generated by BUBBA for each of the 25 tetrominoes alongside the lowest values obtained from

TABLE II. Average lowest-energy configurations for different patchy particles for cMC and LAcMC. $\Delta U = \langle U_{\mathrm{LAcMC}} \rangle / \epsilon - \langle U_{\mathrm{cMC}} \rangle / \epsilon$.

Patchy particle	$\langle U_{\mathrm{cMC}} \rangle / \epsilon$	$\langle U_{\mathrm{LAcMC}} \rangle / \epsilon$	ΔU
1	−3.27 ± 0.04	−3.29 ± 0.03	−0.02 ± 0.05
2	−18.79 ± 0.72	−18.93 ± 0.68	−0.13 ± 0.99
3	−17.13 ± 0.84	−16.10 ± 0.79	1.03 ± 1.15
4	−3.86 ± 0.13	−4.12 ± 0.08	−0.26 ± 0.15
5	17.75 ± 0.56	−17.83 ± 0.52	−0.08 ± 0.76
6	−10.51 ± 0.32	−10.13 ± 0.31	0.38 ± 0.45
7	−20.16 ± 0.63	−19.87 ± 0.59	0.30 ± 0.86
8	−18.82 ± 0.83	−18.61 ± 0.77	0.20 ± 1.13
9	−17.34 ± 0.63	−17.90 ± 0.60	−0.56 ± 0.87
10	11.90 ± 0.36	−11.93 ± 0.35	−0.03 ± 0.50
11	−3.83 ± 0.12	−3.74 ± 0.12	0.09 ± 0.17
12	−19.83 ± 0.56	−19.09 ± 0.60	0.74 ± 0.82
13	−14.35 ± 0.81	−14.19 ± 0.68	0.16 ± 1.05
14	−16.88 ± 0.62	−15.95 ± 0.66	0.93 ± 0.91
15	−11.44 ± 0.32	−11.31 ± 0.32	0.13 ± 0.45
16	−3.84 ± 0.13	−3.74 ± 0.12	0.10 ± 0.18
17	−11.06 ± 0.46	−11.34 ± 0.48	−0.27 ± 0.67
18	−13.58 ± 0.78	−14.08 ± 0.54	−0.50 ± 0.95
19	−18.36 ± 0.73	−18.42 ± 0.75	−0.06 ± 1.05
20	−10.34 ± 0.44	−10.54 ± 0.40	−0.20 ± 0.60
21	−4.19 ± 0.09	−4.28 ± 0.04	−0.09 ± 0.10
22	−16.93 ± 0.61	−17.30 ± 0.58	−0.37 ± 0.84
23	−17.72 ± 0.66	−18.06 ± 0.69	−0.34 ± 0.96
24	−21.25 ± 1.02	−20.57 ± 0.91	0.68 ± 1.37
25	−17.90 ± 0.72	−18.37 ± 0.77	−0.46 ± 1.05

TABLE III. Average number of time steps (in thousands) to find the lowest-energy structures in cMC and LAcMC simulations. $\Delta ts = \langle ts_{\mathrm{LAcMC}} \rangle - \langle ts_{\mathrm{cMC}} \rangle$.

Patchy particle	$\langle ts_{\mathrm{cMC}} \rangle$	$\langle ts_{\mathrm{LAcMC}} \rangle$	Δts
1	255 ± 130	265 ± 113	10.2 ± 172.2
2	378 ± 97	379 ± 98	1.2 ± 138.0
3	380 ± 97	298 ± 117	−82.6 ± 152.1
4	360 ± 103	398 ± 76	38.3 ± 128.3
5	382 ± 105	374 ± 101	−8.4 ± 145.4
6	380 ± 103	254 ± 126	−125.8 ± 162.7
7	407 ± 85	391 ± 90	−16.4 ± 123.4
8	416 ± 75	419 ± 72	2.7 ± 104.4
9	414 ± 85	428 ± 65	14.0 ± 106.9
10	391 ± 95	400 ± 94	8.0 ± 134.0
11	336 ± 112	324 ± 120	−11.9 ± 164.8
12	404 ± 86	347 ± 111	−57.5 ± 140.9
13	412 ± 88	401 ± 77	−11.8 ± 116.5
14	396 ± 94	348 ± 127	−47.7 ± 158.4
15	399 ± 90	380 ± 103	−19.7 ± 137.1
16	359 ± 109	355 ± 113	−4.3 ± 157.5
17	432 ± 66	437 ± 64	5.4 ± 91.7
18	425 ± 76	423 ± 65	−1.9 ± 99.6
19	413 ± 77	427 ± 69	13.4 ± 103.4
20	403 ± 93	417 ± 74	14.4 ± 119.4
21	338 ± 106	354 ± 94	16.1 ± 141.6
22	410 ± 84	408 ± 74	−2.6 ± 112.0
23	393 ± 91	384 ± 88	−8.5 ± 126.3
24	400 ± 92	372 ± 103	−28.1 ± 138.7
25	416 ± 74	416 ± 70	−0.3 ± 101.6

cMC and LAcMC. For all patchy particles except 1 and 21, BUBBA finds the lowest energy configurations of 40 particles. For patchy particles 1 and 21, the best-energy configurations of all three methods are identical. BUBBA also offers a substantial performance improvement in terms of computational time. It takes 4 s to generate the patchy paticle 8 energy-minimizing cluster of size 40 in BUBBA on a 2.8 Ghz iMac. The times for cMC and LAcMC simulations of 5×10^5 time steps are 39.3 and 52.4 s, respectively. Additionally, while a single BUBBA run is an order of magnitude faster than a cMC or LAcMC trial run, hundreds of simulations must be run for the Monte Carlo methods in order to reduce their standard error. This makes BUBBA computationally much more attractive than either MC method for finding potential energy minima, provided that only best-energy or second-best-energy clusters are required to build the best energy structure. This point is discussed in more detail in Sec. V.

V. DISCUSSION

The fundamental differences between MC methods and BUBBA account for the large performance gap we have demonstrated. Finding a configuration of particles that minimizes a system's potential energy is an optimization problem. MC methods are ill suited for optimization problems for two reasons. First, at low temperatures the trial move transition probabilities become diminishingly low. Second, the number of energy-minimizing configurations for a given system can be as low as one. For the systems we consider here, the number of energy-minimizing configurations depends on the shape and patterning of the building blocks. For example, there are two ways to put two copies of patchy particle 11 together to create a cluster with the minimum energy [Fig. 5(a)]. Patchy particle 12, however, only has one energy-minimizing configuration of two particles [Fig. 5(b)]. As the clusters grow larger, the degeneracy of configurations increases for the homogenous particles (1, 6, 11, 16, and 21), but may remain flat for certain other particles (e.g., patchy particle 8 only has one energy-minimizing configuration for every cluster of size $s < 48$).

Optimization problems, such as the one we consider here, can be classified as "easy" or "difficult" depending on the optimal solution degeneracy. Mertens proved that the number of solutions depends on the complexity of the building blocks.[15] Mertens' proof uses a mapping of the number partitioning problem (NPP) onto an Ising model. Using this mapping, it has been shown that the NPP has two regions: a so-called easy region where an instance of the problem has many optimal solutions and a difficult region in which there is only one solution.[15,16] The goal of the NPP is to separate a list of integers into two lists with the smallest difference in sum. If all of the numbers in the list are equal, the problem is trivial: two lists of closest size will have the smallest difference. A list of numbers with widely varying values, however, is much harder to solve. The transition between easy and difficult regions depends on the ratio m/n, where m is the number of bits needed to encode a number in the list and n is the size of the list. When $m/n < 1$ there are many optimal solutions to the problem, otherwise there is only one in the limit of large n. While our systems cannot be mapped easily

TABLE IV. Lowest energy configurations generated with cMC, LAcMC, and BUBBA for 40 identical copies of each particle in Fig. 1.

Patchy particle	U_{cMC}/ϵ	U_{LAcMC}/ϵ	U_{BUBBA}/ϵ
1	−3.35	−3.35	−3.35
2	−21.18	−20.80	−22.40
3	−19.80	−18.98	−20.10
4	−4.20	−4.30	−4.35
5	−19.40	−19.40	−21.15
6	−11.38	−11.15	−11.85
7	−22.58	−21.53	−24.70
8	−21.33	−20.78	−23.40
9	−19.50	−19.95	−22.15
10	−12.95	−12.95	−14.13
11	−4.13	−4.08	−4.25
12	−21.73	−21.03	−23.35
13	−16.68	−16.05	−18.05
14	−19.03	−18.53	−20.88
15	−12.43	−12.40	−13.33
16	−4.23	−4.10	−4.35
17	−12.30	−12.63	−14.10
18	−16.10	−15.98	−18.03
19	−20.20	−20.60	−25.68
20	−11.80	−12.15	−12.83
21	−4.33	−4.35	−4.35
22	−18.43	−18.80	−22.38
23	−19.60	−19.95	−22.83
24	−24.95	−23.33	−29.80
25	−20.13	−20.73	−23.40

onto the NPP, we may expect similar trends in optimization problem complexity. That is, we should expect the patchy particles requiring the fewest bits to encode (particles 1, 4, 11, 16, and 21) to fall into an easy regime relative to the other more complex building blocks. We would therefore expect the MC methods to find energies closest to those found by BUBBA for shapes 1, 4, 11, 16, and 21 while performing relatively worse for all other shapes. Table IV shows that the relative energy differences between the MC methods and BUBBA are indeed closest for shapes 1, 4, 11, 16, and 21, which is consistent with the easy-difficult heuristic of Mertens.

BUBBA is effective relative to MC methods because it checks a smaller number of configurations. In d dimensions, a cluster made up of n subunits has at most $2dn$ perimeter cells. Therefore, when a cluster of n_1 subunits is combined with a cluster of n_2 subunits to form the best energy cluster with $N=n_1+n_2$ subunits, the number C of configurations checked is

$$C = 2dn_1n_2 \tag{2}$$

$$= 2d(N-n_2)(N-n_1) \tag{3}$$

$$= 2d(N^2 - Nn_1 - Nn_2 + n_1n_2), \tag{4}$$

giving

$$O(C) = O(N^2). \tag{5}$$

There are at most $N/2$ pairs of clusters that must be checked to find the best-energy cluster of size N. Also, all of the clusters up to size $N-1$ must be generated before the best-energy cluster of size N can be found. There are $N-2$ such clusters, with sizes 2 through $N-1$. Performing the $O(N^2)$ cluster pairing $N(N-2)/2$ times results in the total number of configurations on the order $O(N^4)$ regardless of volume or the number of spatial dimensions. While a cMC simulation does not attempt to sample all of the $O(V^{dN})$ possible system configurations (N particles in volume V with d spatial dimensions), the fact that BUBBA generates an energy-minimizing cluster after only $O(N^4)$ potential energy calculations shows how tiny a portion of configuration space it samples. By not checking every possible configuration, however, BUBBA is not guaranteed to find the global energy minimum for a set of particles. BUBBA becomes computationally inefficient when generating clusters larger than $N \cong 150$ on a single-processor machine, but we find that the unit cells comprising the energy-minimizing infinite structure can be found when $N<50$ for the family of particles studied here.

FIG. 5. (a) Best energy clusters of size 2 for patchy particle 11. (b) Best energy cluster of size 2 for patchy particle 12.

For the implementation of BUBBA we present here, it is guaranteed to fail if the best cluster of size s is formed by two clusters whose energies are not minima for their sizes. At the expense of computational cost, clusters with suboptimal energies can easily be included in the search for energy-minimizing clusters. We find that in most cases, BUBBA performs well when including only energy-minimizing clusters: generating optimal clusters for 24 of the 25 patchy particles in Fig. 1. Only patchy particle 6 is improved by including second-best clusters. In this case, the best energy $N=4$ cluster is made from two copies of the second-best energy $N=2$ cluster (Fig. 6). By including the second-best clusters for patchy particle 6, BUBBA generates a configuration with $U/\epsilon=-13.3$, lower than that found when only the best-energy clusters are included (Fig. 6).

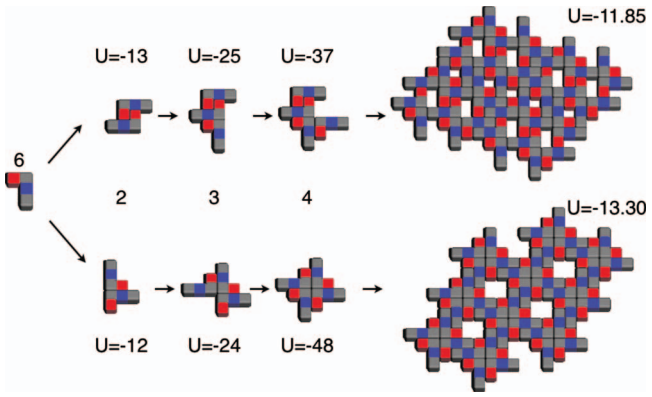

FIG. 6. The best energy $N=4$ cluster for patchy particle 6 can be generated from sub optimal clusters of size 2 or 3.

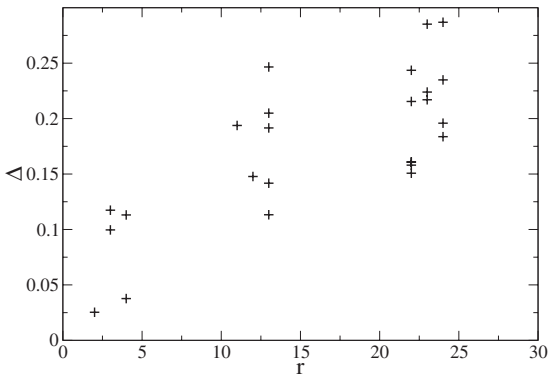

FIG. 7. Normalized difference in energy, $\Delta=(U_{\mathrm{BUBBA}}-\langle U_{\mathrm{cMC}}\rangle)/U_{\mathrm{BUBBA}}$, as a function of strongest relative two-particle interaction energy, $r=U_{\mathrm{strong}}/U_{\mathrm{weak}}$. Multiple shapes can have identical r values.

Given a system of particles at constant temperature, the stronger the interaction energies between two building blocks, the more likely there are to be trial moves with near-zero acceptance probabilities in any MC simulation. For example, in a simulation of shape 12 with $1/\beta\varepsilon=1.0$, the probability of breaking apart the cluster in Fig. 5(a) is $\exp(-22/1.0)=2.9\times10^{-10}$. If such a trial move was required to rearrange a nonoptimal cluster, it would be highly unlikely over the course of a 1×10^{9} time step simulation. Furthermore, this low acceptance probability is approaching the precision of the random number generator, 2^{-b}, where $b=31$ in our simulations.[17] We find that the stronger the interaction energy between two building blocks of a given shape relative to the weakest interaction between them, the more likely they are to become trapped. Here, we quantify a system's susceptibility of becoming trapped by comparing the average minimum potential energies found with cMC in Table II to those found by BUBBA. Figure 7 shows that larger r corresponds to larger performance gaps between BUBBA and cMC. This suggests that another factor contributing to BUBBA's efficacy is the fact that MC simulations of patchy particles can easily become trapped in metastable configurations. Further, the ease with which a MC simulation of patchy particles can become trapped as measured by $\Delta=[(U_{\mathrm{BUBBA}}-\langle U_{\mathrm{cMC}}\rangle)/U_{\mathrm{BUBBA}}]$ appears to increase as r increases.

VI. CONCLUSIONS

Systems of patchy particles that have a hierarchy of interaction energies are complicated, limiting the extent to which traditional simulation methods can be used to predict ordered configurations arising via self-assembly. In this work we have demonstrated that cluster MC and LAcMCs methods are relatively poor methods for finding potential energy minima formed from patchy particles with disparate interaction energies due to their tendency to become trapped in metastable configurations as well as the low degeneracy of potential energy-minimizing configurations. We have shown that LAcMC offers insignificant performance improvements over cMC, but stress that this does not invalidate the potential usefulness of such methods. We have developed a new method—BUBBA—that effectively searches a subset of configuration space for energy-minimizing configurations. While BUBBA, as well as any other heuristic method, is not guaranteed to find optimal solutions, we demonstrate its efficacy relative to cMC and LAcMC. Due to BUBBA's ability to quickly generate energy-minimizing configurations of particles, we expect it to prove useful when many different particles must be evaluated for self-assembly "propensity."[18] Evaluating BUBBA as a screening tool for families of particles such as those studied here is an area of ongoing research.[18]

ACKNOWLEDGMENTS

We thank the Center for the Study of Complex Systems at the University of Michigan for use of their computational resources. We especially thank Mark Ratner and Alessandro Troisi for discussions of their earlier work. Funding for E.J. provided by the NDSEG fellowship and S.C.G. acknowledges support from the James S. McDonnell Foundation.

APPENDIX A: DETAILED BALANCE-OBEYING CLUSTER MOVES

The transitions from microstate to microstate over the course of a MC simulation obey detailed balance if

$$P(o)\pi(o\rightarrow n)=P(n)\pi(n\rightarrow o), \tag{A1}$$

where $P(\alpha)=e^{-\beta U_\alpha}/Z$ is the probability of being in microstatestate α, $\beta=1/k_bT$ is the dimensionless temperature, Z is the partition function, and U_α is the potential energy of microstate α. Obeying detailed balance ensures that if the simulation is not at equilibrium it moves there and remains there once equilibrium has been achieved. In our cMC code, Eq. (A1) expands to

$$\frac{e^{-\beta U_o}}{Z}R_{\mathrm{gen}}(o\rightarrow n)R_{\mathrm{acc}}(o\rightarrow n)=\frac{e^{-\beta U_n}}{Z}R_{\mathrm{gen}}(n\rightarrow o)R_{\mathrm{acc}}(n\rightarrow o), \tag{A2}$$

where $R_{\mathrm{gen}}(\alpha\rightarrow\beta)$ is the rate of generation of a trial move and $R_{\mathrm{acc}}(\alpha\rightarrow\beta)$ is the probability accepting that trial move. Rearranging,

$$\frac{R_{\mathrm{acc}}(o\rightarrow n)}{R_{\mathrm{acc}}(n\rightarrow o)}=e^{-\beta(U_n-U_o)}\frac{R_{\mathrm{gen}}(n\rightarrow o)}{R_{\mathrm{gen}}(o\rightarrow n)}. \tag{A3}$$

Generating a trial move involves selecting a particle at random for use as a cluster seed, generating a cluster from the seed, and selecting a move at random. In every state, the probability of any particle being chosen as the seed particle is simply $1/N$ and the probability of choosing a particular move is $\frac{1}{6}$. As these two factors do not depend on the state, Eq. (A3) reduces to

$$\frac{R_{\mathrm{acc}}(o\rightarrow n)}{R_{\mathrm{acc}}(n\rightarrow o)}=e^{-\beta(U_n-U_o)}\frac{P^n_{\mathrm{gen}}(C|s)}{P^o_{\mathrm{gen}}(C|s)}, \tag{A4}$$

where $P^\alpha_{\mathrm{gen}}(C|s)$ is the probability of generating cluster C

in state α given the seed particle s. Cluster generation begins by defining a cluster C based on seed particle s and calling a function we name NextInCluster that is defined in Algorithm 2.

Algorithm 2 Function NextInCluster(seed s, Cluster C) adds particles to a cluster in a depth-first search tree

Require: seed s, Cluster C
  for all particles l that shares an edge with particle s **do**
    if l has not been added to a cluster **then**
      Add l to cluster C with probability $p_{s,l}$
      if l is added to C and the size of $C < S_{max}$ **then**
        NextInCluster(l, C)

Here, the probability $p_{s,l}$ is the probability that particles s and l are part of the same cluster. This is also known as the bond probability. We define $p_{s,l}=0.5$ if particles s and l share at least one edge and 0 otherwise. Recursively adding neighboring particles to a cluster in this way can generate very large clusters in dense areas. This, combined with the fact that cluster moves can be initiated from any particle means that large clusters are more likely to be selected in dense systems. To account for this, we terminate the growth of a cluster once it has reached a maximum size $S_{\max}=1/\text{rand}$, where rand is a random number chosen uniformly between 0 and 1. By sequentially growing clusters using the method mentioned above, the probability of creating a cluster C given seed particle s and the path chosen over its neighbors is the product of the $p_{k,l}$ over the path selected times the product of $(1-p_{k,l})$ over the cluster's interface. However, there are many paths over which cluster C can be grown. Therefore, the probability of generating a cluster becomes

$$P_{\text{gen}}(C|s) = \left(\sum_i \prod_{k,l \in C} p_{k,l} \right) \prod_{\substack{k \in C \\ l \notin C}} (1 - p_{k,l}), \tag{A5}$$

where the sum over i is the sum of all paths that result in formation of cluster C. Note that this sum is the same for cluster C regardless of state, so Eq. (A4) reduces to

$$\frac{R_{\text{acc}}(o \to n)}{R_{\text{acc}}(n \to o)} = e^{-\beta(U_n - U_o)} \frac{\prod_{\substack{k \in C \\ l \notin C}} (1 - p^n_{k,l})}{\prod_{\substack{k \in C \\ l \notin C}} (1 - p^o_{k,l})}, \tag{A6}$$

where $p^\alpha_{k,l}$ is the probability that particles k and l are in the same cluster in state α. We take the minimum of Eq. (A6) and unity to be the acceptance probability for a MC move,

$$\text{acc}(o \to n) = \min\left(1, e^{-\beta(U_n - U_o)} \frac{\prod_{\substack{k \in C \\ l \notin C}} (1 - p^n_{k,l})}{\prod_{\substack{k \in C \\ l \notin C}} (1 - p^o_{k,l})} \right). \tag{A7}$$

[1] S. C. Glotzer and M. J. Solomon, Nature Mater. **6**, 557 (2007).
[2] D. Frenkel and B. Smit, *Understanding Molecular Simulations: From Algorithms to Applications* (Elsevier, New York, 2002).
[3] J. Liu and E. Luijten, Phys. Rev. E **71**, 066701 (2005).
[4] S. Whitelam and P. L. Geissler, J. Chem. Phys. **127**, 154101 (2007).
[5] M. P. Allen and D. J. Tildesley, *Computer Simulation of Liquids* (Clarendon, Oxford, 1987).
[6] M. N. Rosenbluth and A. W. Rosenbluth, J. Chem. Phys. **23**, 356 (1955).
[7] B. Chen and J. I. Siepmann, J. Phys. Chem. B **105**, 11275 (2001).
[8] M. Mura, N. Martsinovich, and L. Kantorovich, Nanotechnology **19**, 465704 (2008).
[9] A. Troisi, V. Wong, and M. A. Ratner, Proc. Natl. Acad. Sci. U.S.A. **102**, 255 (2005).
[10] C. Domb, J. Phys. A **9**, 10 (1976).
[11] A. Troisi, V. Wong, and M. V. Ratner, J. Chem. Phys. **122**, 0 (2005).
[12] J. Liu and E. Luijten, Phys. Rev. Lett. **92**, 035504 (2004).
[13] R. Swendsen and J. Wang, Phys. Rev. Lett. **57**, 21 (1986).
[14] V. I. Manousiouthakis and M. W. Deem, J. Chem. Phys. **110**, 2753 (1999).
[15] S. Mertens, Theor. Comput. Sci. **265**, 79 (2001).
[16] C. Borgs, J. Chayes, and B. Pittel, Proceedings of the 2001 ACM Symposium on the Theory of Computing, 2001 (unpublished), pp. 330–336.
[17] R. M. Ziff, Comput. Phys. **12**, 385 (1998).
[18] E. Jankowski and S. C. Glotzer (unpublished).