Self Assembly Problems of Anisotropic Particles in Soft Matter

by

Carolyn Louise Phillips

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Applied Physics and Scientific Computing)
in The University of Michigan
2012

Doctoral Committee:

Professor Sharon C. Glotzer, Chair
Professor Leonard M. Sander
Professor Michael J. Solomon
Professor Quentin F. Stout
Professor Robert M. Ziff

ACKNOWLEDGEMENTS

I would like to thank the DOE Computational Science Graduate Fellowship under DOE Grant DE-FG02-97ER25308, and specifically the staff of people who manage and advise this fellowship. I deeply appreciate that you do not just fund graduate students but actually develop them. Thanks for your occasional redirections, for tolerating the early flailing of my PhD, and for your unflagging support. I would also like to also thank Department of Energy, Office of Science, Basic Energy Sciences, (DE - FG02 - 02ER4600) for funding me my last year and a half of my PhD.

I would like to acknowledge the incredibly talented and diverse Glotzer Group. As "steel sharpens steel," I have treasured my interactions with this great group of minds. Specifically, Eric Jankowski who lured me to the the Glotzer Group with his discussions of emergence, complex system and tetris pieces. It only took us four years to find a project to work on together! Joshua Anderson, who taught me everything I know on how to really write code. Aaron Keys, Michael Engel and Amir Haji-Akbari who were essential in molding my thinking at critical moments. Chris Iacovella, my first mentor, and Ryan Marson, who is carrying on the cause.

I would like to thank my advisor Sharon Glotzer, who creates a resource-rich creative environment for her research group. I would like to officially recognize that you were right more often than this stubborn researcher was willing to acknowledge on many occasions. I would like to thank my dissertation committee for their support and their willingness to wade through this diverse research work!

I would also like to acknowledge my wife, Heather, who supported me, encouraged

me, listened to every presentation and always gave the best feedback, talked me up and talked me down. You were the best part of my PhD, are the best part of my life, and I promise to clean up the living room when this is all over so our house does not become the Ann Arbor Math Museum.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF APPENDICES

Appendix

ABSTRACT

Self Assembly Problems of Anisotropic Particles in Soft Matter

by

Carolyn Louise Phillips

Chair: Sharon C. Glotzer

Anisotropic building blocks assembled from colloidal particles are attractive building blocks for self-assembled materials because their complex interactions can be exploited to drive self-assembly. In this dissertation we address the self-assembly of anisotropic particles from multiple novel computational and mathematical angles.

First, we accelerate algorithms for modeling systems of anisotropic particles via massively parallel GPUs. We provide a scheme for generating statistically robust pseudo-random numbers that enables GPU acceleration of Brownian and dissipative particle dynamics. We also show how rigid body integration can be accelerated on a GPU. Integrating these two algorithms into a GPU-accelerated molecular dynamics code (HOOMD-blue), make a single GPU the ideal computing environment for modeling the self-assembly of anisotropic nanoparticles.

Second, we introduce a new mathematical optimization problem, *filling*, a hybrid of the familiar shape packing and covering problem, which can be used to model shaped particles. We study the rich mathematical structures of the solution space and provide computational methods for finding optimal solutions for polygons and convex polyhedra. We present a sequence of isosymmetric optimal filling solutions for

the Platonic solids. We then consider the filling of a hyper-cone in dimensions two to eight and show the solution remains scale-invariant but dependent on dimension.

Third, we study the impact of size variation, polydispersity, on the self-assembly of an anisotropic particle, the polymer-tethered nanosphere, into ordered phases. We show that the local nanoparticle packing motif, icosahedral or crystalline, determines the impact of polydispersity on energy of the system and phase transitions. We show how extensions of the Voronoi tessellation can be calculated and applied to characterize such micro-segregated phases. By applying a Voronoi tessellation, we show that properties of the individual domains can be studied as a function of system properties such as temperature and concentration.

Last, we consider the thermodynamically driven self-assembly of terminal clusters of particles. We predict that clusters related to spherical codes, a mathematical sequence of points, can be synthesized via self-assembly. These anisotropic clusters can be tuned to different anisotropies via the ratio of sphere diameters and temperature. The method suggests a rich new way for assembling anisotropic building blocks.

CHAPTER 1

Introduction

1.1 Nanotechnology

Nanotechnology is a very old technology. The functioning of all biological life has been based on the successful application of nanotechnology for at least 3.5 billion years. We humans are late in entering the field and have quite some work ahead of us to begin to compete with the beautiful and intricate inventions of Nature. What we have to our advantage is the directed application of knowledge and inventiveness as compared to the slow stumbling advances of evolution.

humans have been manipulating the nanoscopic properties of materials since the bronze age. The Japanese artisans who manufactured samurai swords in the 6th century, for example, were masters of metallurgy with elaborate techniques for create masterpieces of composite steels[8]. Their methods, however, were passed down as sacred arts, without a true understanding of how and why each part of the process worked. The human adventure into nanotechnology only truly began in the last century, when we finally gained the capacity to peer inside materials with the discovery of atomic theory and the invention of progressively more powerful microscopes.

The potential of nanotechnology can be gauged by the diversity of ordered structures, life, and almost-life that can be found at scales smaller than 10^{-6} meters, from DNA to viruses to bacterium. As proclaimed by physicist Richard Feynman in a talk

in 1959, "There's plenty of room at the bottom" [9]. The human body is 10^{10} times larger than a hydrogen atom. The Earth, in comparison, is only 10^7 times larger than a human. And on one third of the surface of our plane we fit houses, economies, and nations. A grain of sand contains an enormous amount of space for structure and design.

Gaining engineering control over these microscopic scales is a technological challenge. In the same talk, Feynman offered a $1000 prize to the first person who could construct an operating electric motor that fit in a 1/64 inch cube. In this case, the prize was collected only one year later by William McLellan, who, over two and half months, constructed the tiny motor using a microscope, a watchmakers lathe and a toothpick.

However, rather than assembling microscopic to nanoscopic structures by laborious direct manipulation of atoms, molecules, nanoparticles, colloidal particles, or *top-down assembly*, it has been proposed to let the particles do the work for you. Instead of optical tweezers or mask lithography, it has been proposed to synthesize nano-scale building blocks that assemble into structures spontaneously, motivated by the driving forces of equilibrium and non-equilibrium thermodynamics, or *bottom-up assembly* or *self-assembly*. However, this method presumes, first, the ability to design and synthesize cooperative nanoparticles, and second, knowledge of the "recipes" for different targeted structures.

Glotzer and Solomon [10] point out that nanoparticle synthesis is becoming maturing powerful tool and, as a result, there is now an unprecedented diverse spectrum of particle types to consider. They characterize these building blocks by different axes of *anisotropy* or design features that differentiate a particle from a simple isotropic sphere. The axes of particle anisotropy include the fraction of the surface area covered by patch material, aspect ratio, faceting, interaction patchiness via surface pattern quantization, branching, chemical ordering, shape gradient, particle roughness, and

chirality. These anisotropic designed particles become the "ingredients" of material design and assembly. This vast menagerie of anisotropic particle possibilities has grown beyond analogy to the periodic table of elements and its mere 118 elements.

With each anisotropic degree of freedom, the dimensionality of the design space of nanoparticles increases. Also whole new avenues of creative design become possible that previously had no physical realization. The fundamental questions of nanoscale self-assembly thus range from the thermodynamics of self-assembly to the engineering questions of designed materials. How can novel ordered structures be self-assembled thermodynamically from designed nanoscale building blocks? Can we discover design rules for the engineering of new materials? Can we make fast and reliable predictions?

The field of nanoscale self assembly exists at the nexus of soft matter physics, chemistry, complexity theory, computational geometry, computational science, and material science. In this work we address questions above primarily through the tool of computer simulation.

1.2 Computer Simulation

The two traditional paradigms of scientific investigation are theory and observation/experiment where science advances due to the fruitful cooperation and tension between the two. Computational simulation of physical systems is now considered to be the third paradigm of scientific investigation. Computational simulation can provide unique insights when the system is too small (atoms), too large (galaxies), too dangerous (nuclear explosions), or too expensive (nuclear reactors) to directly physically create in experiment. Simulations also become important contributors when the parameter space is too large to directly physically explore (material discovery) or when the emergent behavior of a simply described system cannot be discovered by mathematical analysis alone (complex fluids). In investigating nanoscale self-assembly, computer simulation is useful to explore the connection between the simple

agents, rule sets, or building blocks and the emergent larger structure.

The computational method used primarily in this dissertation is Molecular Dynamics. Molecular Dynamics is an N-body method[11] whereby a population of particles is simulated moving in space following Newtonian trajectories due to the force interactions between the particles. Molecular Dynamics simulations of particles at equilibrium can be used to sample ensembles of the particles and extract statistically relevant properties. In 1964, Aneesur Rahman, the father of molecular dynamics, modeled 864 liquid Argon atoms for a simulated time length of 10^{-11} seconds on a CDC 3600 capable of 150,000 calculations per second, or 150 kiloflops. Argon atoms were modeled by a truncated Lennard-Jones pair potential[12]. The success of this simulated system in modeling the properties of a real Argon fluid is governed by how well the essential physics of interacting Argon atoms is modeled, the number of atoms, and the time length of the simulation. Even in this first simulation experiment, good agreement was found between the simulation and experimentally observed values. The Lennard-Jones pair potential used by Rahman is a computationally efficient potential for modeling a hard core repulsive interaction and long range attraction, emulating Pauli repulsion and Van der Waals interaction of atoms. As a result the properties of a fictitious material called "Lennard-Jonesium", have become an independent topic of study as a computational surrogate for a range of possible physical systems.

The field of computational simulation has, of course, been inextricably linked to growing capability of computer technology. Driven by Moore's "Law" regarding the density of transistors in silicon, computational speed and memory has been improving at roughly exponential rates since the middle of the 20th century. The heroically massive computations of one generation are performed on the personal computers of the next. And as length and size of simulations have improved, so has the understanding the simulated systems being. For example, at one time a rapidly cooled and uncrystallized fluid of Lennard-Jonesium was considered to be a good model for

a structural glass. In 1998, Schneidman and Uhlmann wrote a letter entitled "Does a Lennard-Jones glass exist?" In this letter they identified that the theory that any physical liquid could turn into a structural glass if quenched sufficiently rapidly had been falsely supported by simulations.

> *The existing data which support the formation of an LJ glass are mainly based on monitoring the structure of an undercooled fluid for times of the order of* $10^{-12}-10^{-11}$ *s. Unfortunately, this provides very little information whether the fluid will remain a glass on a laboratory time scale*[13].

While glasses are still a complicated and mysterious topic in physics, it is no longer thought that any fluid can form one, and single-component Lennard-Jonesium is certainly no longer considered to be a reasonable glass model*.

One of the powerful uses of Molecular Dynamics (MD) is its ability to model objects much larger than atoms. The classical Newtonian physics at the core of MD does not care what "particles" are as long as the forces acting on them can be expressed mathematically and in a continuously differentiable manner. Molecular Dynamics has been used to model various types of complex fluids such as granular flow[15], liquid crystals[16,17], polymers[18], colloidal suspensions[19–21], and even theoretical models of swarms[22,23].

The Glotzer research group, where research focuses on understanding the emergence of ordered structures in soft matter and nanoscale systems, has used molecular dynamics to model tethered nanospheres[24–26], nanorods[5,27,28], and plates[29], and shape shifting nanoparticles[30].

The Argon atoms of the first MD computer experiment are now anisotropic colloidal particles, but the basic needs have remained the same. How can a simulation experiment be performed with current computing technology so that results on sufficiently sized systems can be achieved in a reasonable amount of researcher time? How can the "atoms" be modeled so that the essential physics of interaction is captured?

*binary component Lennard Jonesium, composed of two types of LJ particles, is another matter[14]

And then, the new question that has arisen, what if you can design the "atom"? What new mesoscopic ordered structures can emerge?

This dissertation is organized into three parts addressing three questions:

I. How can current computing technology be used to make simulations faster?

II. How can new types of building blocks be modeled?

III. How can building blocks be designed to self-assemble into novel structures?

1.3 How Can Current Computing Technology Be Used to Make Simulations Faster?

In modern computing, the current paradigm for accelerating computation is not improving processor speed, but parallelizing computation over many processors or cores[11]. The challenge of running large computations is determining the optimal way to parallelize the computation so that work can be allocated over many resources. A decade ago, "massively parallel" computations implied clusters of thousands of CPUs that were available at large government laboratories or large companies. However, fueled by the video game industry and the demand to render physically realistic video game images in real time, companies such as NVIDIA, Intel, and ATI, manufactured commodity specialized processors that perform massively parallel computation, namely graphics cards or GPUs.

As these devices became programmable and as accessible programming languages such as CUDA™and OpenCL became available to the general research community, these devices brought affordable massive parallelization to the academic and small business communities.

In turn, they have influenced the world of large-scale super computing. At the time of the writing of this dissertation, three of the top five fastest supercomputers in

the world are hybrid GPU/CPU clusters (Tianhe-1A, Nebulae, and TSUBAME 2.0), with more, even faster, hybrid systems being planned to be operational in the next year (ORNL's "Titan" using the NVIDIA Kepler GPU, predicted to peak at 10-20 petaflops.).

However, designing calculations and algorithms to be massively parallelized so that they can be implemented on these specialized computing devices is far from trivial. In the first part of this dissertation, we review the algorithms and methods of Molecular Dynamics as used in this research in chapter and the basics of the GPU parallelization paradigm in chapter 2,, and work done in building the capability of GPU-accelerated MD for modeling anisotropic nanoparticles in chapters 3 and 4.

In chapter 3 we discusses how a rigid body constraint algorithm is incorporated into HOOMD-blue, a massively parallel GPU-accelerated MD application. Two case studies are presented in this chapter where HOOMD-blue consistently executes a factor of 2.5–3.6 times faster than the peak performance of the LAMMPS code package[31] parallelized over any number of cores. The HOOMD-blue software is primarily developed by Dr. Joshua Anderson, in collaboration with myself, Dr. Trung Nguyen, and Dr. Aaron Keys, as well as other contributors from around the world. The work of this chapter was done in collaboration with Dr. Joshua Anderson and Dr. Trung Nguyen.

Two extensions of MD used to model soft matter at colloidal scales are Brownian dynamics (BD) and dissipative particle dynamics (DPD). In chapter 4, we discuss the generation of massively parallel pseudorandom numbers to support BD and DPD simulations in HOOMD-blue. By deploying a novel scheme for generating pseudorandom numbers on the GPU, the force calculation step of BD is made modestly faster (10%) but the force calculation step of DPD is made five times faster. To date, HOOMD-blue is still the only MD code package with true GPU-accelerated DPD. This work was done in collaboration with Dr. Joshua Anderson.

1.4 How Can New Types of Building Blocks Be Modeled?

In Part II of this dissertation, we introduce a method for optimally generating the volume-excluding shape in molecular dynamics simulations.

In an MD simulation, one way to model a "shaped" particle is to treat it as a body composed of a rigidly connected set of points, where each point has an isotropic pair potential interaction with all the points in other bodies. If the pair potential is solely repulsive, with a rapid transition from infinite to zero and a tunable distance of interaction, the body is essentially a set of hard spheres, or balls, (that may or may not overlap) of varying diameters rigidly bonded together.

While in MD, many aspects of a simulation contribute to the computational performance, in general it is desirable to model a system with as few point particles as possible. A natural question to ask, then, is

Given a "shaped" particle one wishes to model with N point particles that act as overlapping balls, what is the best arrangement of the points to represent the shape?

We propose that the best arrangement is the one that has the maximal *filling* value, where filling is defined as the fraction of the volume of the shape the spheres cover, without any of the spheres crossing the boundary of the shape. We now can think of the shape, itself, as a container that we are placing objects inside of in an optimal way so to fill that container.

As so defined, filling is distinctly different from spatial distribution problems such as *packing*, the arrangement of non-overlapping objects inside a container, and *covering*, the arrangement of overlapping shapes whose union contains the container. Filling may be thought of as a hybrid of *packing* and *covering*, where the objects are packed inside the container, but are free to overlap within it.

Chapters 5-7 represent the first description of this novel mathematical problem. The properties of optimal solutions are explored and true optimal solutions are constructed by multiple numerical methods.

In chapter 5 we consider the question above in an N-dimensional space, and derive universal properties of filling solutions. In chapter 6, we consider the properties of a filling solution in 2D. For polygons, we construct a heuristic algorithm capable of generating optimal solutions, which is compared to a genetic algorithm authored by Dr. Joshua Anderson. We also derive the distribution of discs in the continuum solution ($N \to \infty$) for a polygon. In chapter 7, we show an algorithm that can generate filling solutions in three dimensions for polyhedra. We then discuss the interesting properties of the filling solutions found for the platonic solids in three-dimension and the isosymmetric solutions where the filling solution has the same symmetry as the Platonic solid.

This work was performed in collaboration with Dr. Joshua Anderson, who developed the genetic algorithm, and Dr. Beth Chen, who assisted in the creation of the phase diagrams of the reduced three-disc problem and derived the hypercone equations. Both also provided invaluable contributions discussing and reviewing the correctness of the mathematical content. We also acknowledge useful discussions with Dr. Greg Huber of the University of Connecticut at the initiation of this project regarding the novelty of the mathematical question.

1.5 How can building blocks be designed to self-assemble into novel structures?

In Part III, we use the methods and software discussed in Part I to study the self-assembly for specific anisotropic particle systems.

In chapter 8, we consider how nanoparticle polydispersity impacts the self-asembly of ordered phases of anistropic nanoparticles. One difference between a system of Argon atoms and a system of anisotropic nanoparticle "atoms" is that while in the former case, every atom is physically identical to every other atom, in the latter case,

this is never exactly true under even the most stringent manufacturing conditions. Each "atom" is composed of thousands to millions of physical atoms and no two is composed precisely identically. If the anisotropic building blocks are predicted to self-assemble into a mesoscopic structure when they are identical, how much variation in the building block can be tolerated?

In this work, we focus primarily on the variation in shape of the building block. The building block is a polymer-tethered nanosphere, which simulation predicts to assemble into hexagonally packed cylinder, double gyroid, perforated lamallae, and crystalline lamellar bilayers[24,25]. We study the impact of polydispersity, or variation in the nanosphere diameter, on these phases. In general we show that the more ordering in the nanosphere domain of the phase, the more sensitive the phase is to polydispersity. This study was done in collaboration with Dr. Chris Iacovella.

In chapter 9 we discuss a method for measuring the packing density of a subregion of a microsegregated phase, such as what is formed by the polymer-tethered nanosphere. By performing a Voronoi tessellation on a snapshot of the system, the aggregate packing fraction of the Voronoi cells associated with a single component can be measured. In systems composed of or modeled by spheres of bidisperse or polydisperse diameters, the Voronoi tessellation, which does not take into account diameters, can generate conflicting results. Instead an extension to the Voronoi tessellation in the form of a Voronoi S tessellation, or a radical tessellation[32,33] (also referred to as a power diagram[34,35] or Laguerre tesselation[36]) is needed. We show how these tessellations can be calculated by extending a plane-cutting Voronoi cell method[37]. While the Voronoi S tessellation is a more physically intuitive tessellation, the radical tessellation shows the same density trends as a function of polydispersity and is far more computationally efficient to calculate.

In chapter 10, we study small clusters of two types of particles that can be self-assembled to create anisotropic particles. Colloidal anisotropic particles are attractive

building blocks for self-assembled materials because their complex interactions can be exploited to drive self-assembly. We show how clusters can be robustly self assembled from simple spherical particles of controllable diameters. Clusters of three to thirteen such spheres are self-assembled with *spherical code*[38] point arrangements around one of the spheres at a range of temperatures. These clusters are studied using both molecular dynamic and free energy methods. The method introduce for assembling these clusters suggests a rich new way for assembling anisotropic clusters.

This work was done in collaboration with Eric Jankowski, who authored the free energy calculation method. This work was also assisted in its early stages by Michelle Marval, an undergraduate in Material Science and Engineering.

All the work above was done under the advisement of Prof. Sharon C. Glotzer.

CHAPTER 2

Molecular Dynamics Methods

2.1 Molecular Dynamics

Molecular dynamics (MD) simulations and related methods are powerful tools for modeling systems of particles[39]. The basic MD technique computes the trajectory of n particles under the influence of a potential $V(\vec{x}_1, \vec{x}_2, ..., \vec{x}_n)$, the negative gradient of which gives a conservative force $\vec{F} = -\vec{\nabla}V$, by integrating Newton's equations of motion over discrete time steps that each advance the state of the system from $[\vec{x}_i(t), \vec{v}_i(t)]$ to $[\vec{x}_i(t + \Delta t), \vec{v}_i(t + \Delta t)]$. The quantities $\vec{x}_i$ and $\vec{v}_i$ are the position and velocity of the i-th particle, respectively, t is the current simulation time, and Δt is the step size. Classical MD breaks the potential into pair-wise and bond terms $V = \sum_{\text{pairs } i,j} V_{\text{p}}(x_{ij}) + \sum_{\text{bonds } i,j} V_{\text{b}}(x_{ij})$.

We use a velocity-Verlet integration scheme[40] to update the system at each time step. This integration scheme breaks the integration into three substeps.

1. **Initial Integration**

$$\vec{v}_i\left(t + \frac{\Delta t}{2}\right) = \vec{v}_i(t) + \vec{f}_i(t)\,\frac{\Delta t}{2m} \tag{2.1}$$

$$\vec{x}_i(t + \Delta t) = \vec{x}_i(t) + \vec{v}_i\left(t + \frac{\Delta t}{2}\right)\Delta t \tag{2.2}$$

2. **Force Calculation** $\vec{f_i}(t + \Delta t, \vec{x}(t + \Delta t))$

3. **Final Integration**

$$\vec{v}_i(t + \Delta t) = \vec{v}_i\left(t + \frac{\Delta t}{2}\right) + f_i(t + \Delta t)\,\frac{\Delta t}{2m} \tag{2.3}$$

2.1.1 Brownian dynamics

An NVT canonical ensemble is modeled by a thermostat added to the system[40]. The Brownian Dynamics (BD) thermostat models the solvent-colloidal particle interaction by applying a non-momentum conserving Langevin force[40,41]. In BD, a temperature dependent random force and a drag force proportional to the particle velocity is applied to each particle at each time step of the simulation. The force F applied to each spatial component of particle i is

$$F_i = -\gamma v_i + R_i\sqrt{3}\sqrt{\frac{2k_{\mathrm{B}}T\gamma}{\Delta t}} \tag{2.4}$$

where γ is a coupling constant, T is the temperature of the solvent, Δt is the time step size, v_i is the velocity of the particle, k_{B} is the Boltzmann constant, and R_i is a random number uniformly distributed in the range [-1,1].

Simulations using a BD thermostat are referred to as Brownian Dynamics (BD) simulations. We use BD to simulate the self-assembly of particles in chapters 8 and 10.

2.2 Coarse-Grained Pair Potentials and Bonds

2.2.1 Shifted potentials

We use minimal phenomenological coarse-grained molecular models for the interactions between particles. Different models of the potential energy for pairs of particles have certain convenient features for desired minimal models. A primary

interest in this work, however, is modeling the interactions of particles with different diameters. Any of the potentials mentioned above can be modified to handle a collection of particles with different diameters by *shifting* the potentials radially by a distance parameter α_{ij} that is determined for each particle pair. Each of the following potentials has a singularity which is then shifted by α_{ij}. For two particles of modeled diameter σ_i and σ_j, $\alpha_{ij} = (\sigma_i + \sigma_j)/2 - \sigma$, where σ is the unit of length of each potential. Accordingly, the radial range over which each potential is evaluated, or cutoff, must be increased by α_{ij}.

2.2.2 Shifted Lennard-Jones

Shifted Lennard-Jones includes a short range volume excluding interaction component (r^-12) and middle-range attraction interaction component (r^-6) between two particles of diameters σ_i and σ_j. It is modeled using a radially shifted 12-6 Lennard-Jones potential (LJ), which is also truncated and shifted to zero at r_{cutoff}. This potential can be used to model solvent selectivity for nanoparticles for which the solvent is poor, which results in a condition where nanoparticles tend to aggregate.

$$U_{LJS} = \begin{cases} 4\epsilon\left(\left(\frac{\sigma}{r-\alpha_{ij}}\right)^{12} - \left(\frac{\sigma}{r-\alpha_{ij}}\right)^{6}\right) & \\ \quad -4\epsilon\left(\left(\frac{\sigma}{2.5}\right)^{12} - \left(\frac{\sigma}{2.5}\right)^{6}\right) & r < r_{cutoff} \\ 0 & r \geq r_{cutoff} \end{cases} \tag{2.5}$$

where, $r_{cutoff} = 2.5\sigma + \alpha_{ij}$. The cut off of 2.5σ is a commonly chosen cutoff distance where the potential is 1/60th of its minimum value.

2.2.3 Shifted WCA

The purely repulsive Weeks-Chandler-Andersen (WCA) soft-sphere potential[42] is used to model to short-range, excluded volume interactions. This potential can be

used to model solvent selectivity for polymer tethers for which the solvent is good.

This potential can also be used to model "hard" spheres. In this case, by shifting the potential, the interaction between spheres of diameter $(\alpha + \sigma)$ can be made arbitrarily hard.

$$U_{WCA} = \begin{cases} 4\epsilon \left(\left(\frac{\sigma}{r-\alpha_{ij}} \right)^{12} - \left(\frac{\sigma}{r-\alpha_{ij}} \right)^{6} \right) + \epsilon & r < r_{cutoff} \\ 0 & r \geq r_{cutoff} \end{cases} \tag{2.6}$$

where $r_{cutoff} = 2^{1/6}\sigma + \alpha_{ij}$. The cut off of $2^{1/6}\sigma$ truncates the potential at $U_{WCA} = 0$.

2.2.4 Shifted Morse

The Morse potential[43] was originally formulated to model diatomic bonds. However, it can be used to model a variety of coarse-grained particle-particle bonds. A convenience of this potential is that the well depth and width can be explicitly controlled by the parameters E_0 and β, respectively. E_0 has units of ϵ and β has units $1/\sigma$.

$$U_{Morse} = \begin{cases} E_0 \left(e^{-2\beta(r-\sigma-\alpha_{ij})} - 2e^{-\beta(r-\sigma-\alpha_{ij})} \right) & r < r_{cutoff} \\ 0 & r \geq r_{cutoff} \end{cases} \tag{2.7}$$

where $r_{cutoff} = 2.5\sigma + \alpha_{ij}$. The cutoff of 2.5σ is specified here because, as parameterized in this dissertation, the Morse potential has a similar range to the Lennard-Jones potential.

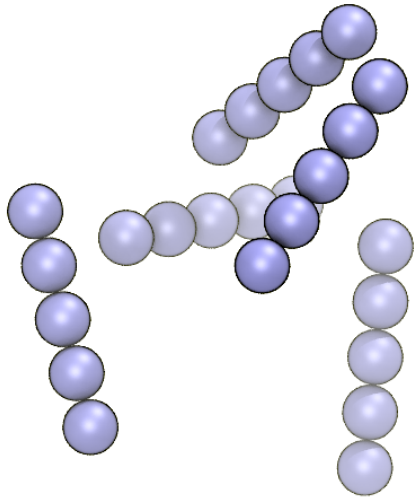


Figure 2.1: A collection of "rods" composed of five beads integrated as a rigid unit.

2.2.5 Shifted FENE

Polymer tethers are modeled as bead-spring chains Bonds between polymer tether beads are modeled via a finitely extensible non-linear elastic (FENE) spring[44].

$$U_{FENE} = -\frac{1}{2}kr_0^2 \ln\left(1 - \left(\frac{r - \alpha_{ij}}{r_0}\right)^2\right) + U_{WCA} \tag{2.8}$$

where r_0 is the minimum distance of the bond.

2.3 Rigid Body Models

One way to model anisotropic nanoparticles in MD is to create a composite body out of point particles with isotropic pair potentials as done by Zhang and Glotzer[2]. Directly rigid composite bodies by connecting point particles via steep, "hard" potentials, such as bonds with stiff spring constants, requires using a prohibitively small step size to maintain accuracy and stability. Potentials with infinitely steep interaction terms can only be achieved with extensions to the basic MD framework.

One such extension is SHAKE[45]. The SHAKE algorithm enforces fixed bond

distances between two particles. Via an iterative method, any number of bonds in the system can be constrained. A set of particles may be combined into a single rigid body with an appropriate choice of bond constraints while taking special care not to over-constrain the system. However, certain rigid shapes, such as planar and linear molecules, cannot be created in three dimensions by setting bond distances alone because the constraint matrix is singular. Although the SHAKE algorithm has been extended to handle arbitrary shapes, for example, via angle and dihedral angle constraints,[46,47] the computational cost of these algorithms often becomes prohibitive for parallel simulation codes as the number of constraints per cluster increases.

Modeling large or generic rigid arrangements of particles can also be achieved by treating each defined set of particles as a single rigid body with only three translational and three orientational degrees of freedom (or two and one, respectively, for 2D simulations)[48]. Such a method can be added to a MD package with minimal modifications by taking advantage of the existing code that computes particle-particle interactions. Rigid body constraints are available in MD software packages such as DLPOLY[49] and LAMMPS[50], and have been used to model cubes, rods, bent rods, jacks, plates, bumpy spheres, water molecules and ions, and Buckyballs.[51–58].

First we introduce the terminology of a system of rigid and non-rigid bodies and then show how rigid body constraints are incorporated into an MD scheme. A system contains n particles, each of which may belong to one rigid body or none at all, for a total of N_{bodies} rigid bodies such that $N_{\text{bodies}} \leq n$. The center of mass and velocity in the *space frame* shall be indicated by lowercase $\vec{r}$ and $\vec{v}$ for a particle and uppercase $\vec{R}$ and $\vec{V}$ for a rigid body with appropriate subscript indices.

Consequently, each rigid body b is composed of N_b particles indexed by $B_{bk} = [B_{b1}, B_{b2}, ...B_{bN_b}]$. The center of mass of body b is located at position $\vec{R}_b$, moving at a velocity $\vec{V}_b$. Body b has a mass M_b and moment of inertia $\mathbf{I}_b$. The orientational degrees of freedom include its angular momentum $\vec{L}_b$ and a normalized quaternion

q_b representing its orientation. In the *body frame*, a body's center of mass is at the origin and $\mathbf{I}_b$ is diagonal.

Thus, the position and velocity of a particle in the space frame can be calculated as follows:

$$\vec{r}_{B_{bk}} = \vec{R}_b + \mathbf{R}(q_b) \cdot \vec{D}_{bk} \tag{2.9}$$

$$\vec{v}_{B_{bk}} = \vec{V}_b + \vec{\omega}_b \times (\mathbf{R}(q_b) \cdot \vec{D}_{bk}), \tag{2.10}$$

where $\vec{D}_{bk}$ is a displacement vector that defines the position of the particle relative to the center of mass (COM) in the body frame and $\vec{\omega}_b = \mathbf{R}(q_b)\mathbf{I}_b^{-1}\mathbf{R}^T(q_b)\vec{L}_b$ is the body's angular velocity about its COM. $\mathbf{R}(q)$ is a 3x3 matrix that rotates vectors from the body frame to the space frame[59].

The net force $\vec{F}$ and torque $\vec{\tau}$ acting on body b in the space frame are the sums of the individual forces and torques resulting from the particle-particle forces $\vec{f_i}$ computed by existing algorithms. The sums

$$\vec{F_b} = \sum_{k=1}^{N_b} \vec{f}_{B_{bk}} \tag{2.11}$$

and

$$\vec{\tau_b} = \sum_{k=1}^{N_b} [\mathbf{R}(q_b) \cdot \vec{D}_{bk}] \times \vec{f}_{B_{bk}} \tag{2.12}$$

are performed over all constituent particles.

2.3.1 NVE integration scheme

In the microcanonical NVE ensemble, Newtonian mechanics[59] governs the motion of rigid bodies with the following equations

$$\dot{\vec{R}}_b = \vec{V}_b \tag{2.13}$$

$$\dot{\vec{V}}_b = \vec{F}_b / M_b \tag{2.14}$$

$$\dot{\vec{L}}_b = \vec{\tau}_b \tag{2.15}$$

$$\dot{q}_b = \frac{1}{2}\mathbf{A}(\vec{\omega}_b) \cdot q_b, \tag{2.16}$$

where $\mathbf{A}(\vec{\omega}_b)$ is a 4x4 matrix defined in reference[59].

These equations are numerically integrated in a way analogous to the velocity Verlet discretization scheme used for unconstrained particles[39]. The velocity and angular momentum of each rigid body are first updated to $t + \Delta t/2$, and the position and orientation are updated to $t + \Delta t$ by the equations

$$\vec{V}(t + \Delta t/2) = \vec{V}(t) + \frac{\Delta t}{2M} \cdot \vec{F(t)} \tag{2.17}$$

$$\vec{R}(t + \Delta t) = \vec{R}(t) + \Delta t \cdot \vec{V}(t + \Delta t/2) \tag{2.18}$$

$$\vec{L}(t + \Delta t/2) = \vec{L}(t) + \Delta t/2 \cdot \vec{\tau(t)} \tag{2.19}$$

$$q(t + \Delta t) = \mathrm{Q}\left(q(t), \Delta t, \vec{L}(t + \Delta t/2), I\right), \tag{2.20}$$

where the function Q is an application of the Richardson method[60].

Forces and torques are then calculated based on the updated positions and orientations, and the velocity and angular momentum are advanced fully to $t + \Delta t$,

$$\vec{V}(t + \Delta t) = \vec{V}(t + \Delta t/2) + \frac{\Delta t}{2M} \cdot \vec{F}(t + \Delta t) \tag{2.21}$$

$$\vec{L}(t + \Delta t) = \vec{L}(t + \Delta t/2) + \frac{\Delta t}{2} \cdot \vec{\tau}(t + \Delta T) \tag{2.22}$$

2.3.2 NVT and NPT integration schemes

One method to model a system of rigid bodies in a canonical NVT ensemble is to combine a Langevin thermostat with an NVE integration scheme, also known as Brownian dynamics (BD)[39]. The thermostat is applied to each individual particle in the system, which effectively thermalizes the rigid bodies without momentum conservation. This thermostat is also not a physically accurate model of the interaction of a solvent with a rigid body, as the random molecular and viscous forces act isotropically on each particle of the body.

Simulations in the NVT ensemble, as well as isothermal-isobaric NPT ensemble, can also be accomplished with the application of a Nosé-Hoover thermostat (and for NPT, a barostat) with an extended Hamiltonian. Miller and coauthors[59] derive a Hamiltonian formulation of the NVE rigid body equations of motion by introducing the conjugate quaternion momentum. Kameraj and coauthors[61] extend it with requisite thermostat and barostat and derive the resulting numerical integration steps similar to Equations 2.17–2.22.

2.4 GPU Computing

In recent years GPUs have become affordable, easily programmable, general-purpose massively parallel processors. Originally designed for rendering graphics, the massively parallel architecture of GPUs makes them well suited for many scientific computing problems. Algorithms that exploit fine-grained parallelism have been accelerated by orders of magnitude including financial models[62], computational fluids dynamics[63], linear algebra performed by the GPU accelerated BLAS, LAPACK, and sparse matrix libraries[64,65], and Fast Fourier Transforms[64,65].

The GPU hardware is unlike a CPU in many ways. First and foremost, while a CPU core executes a single instruction at a time, a GPU executes hundreds. The

processor chip on the NVIDIA® GeForce® GTX 480 (GF100), for example, contains 480 individual CUDA cores. Each core is capable of processing one single precision floating point or integer operation per clock tick. The CUDA cores on the GTX 480 are grouped into 15 multi-processors (MPs), which perform instruction scheduling and are each capable of maintaining up to 1536 independent computation streams or *threads* in flight at any one time. Thus the GPU is only fully occupied when more than 23 thousand threads are executing on the device.

GPUs are Single-Instruction-Multiple-Thread (SIMT) parallel devices. In a GPU environment, algorithms with fine-grained parallelism distribute calculations over thousands of simultaneous threads, each sharing instructions but operating on different data. A *kernel* is a launch of a large group of these threads scheduled in blocks across a number of multiprocessors on the GPU device. Each thread has access to a fraction of the resources of its multiprocessor, can share a small amount of memory with other threads in the same block, but cannot communicate directly with other running threads. GPUs have a hierarchy of memory structures, from global memory, which is large, accessible by all threads and has a high bandwidth but a long latency, to shared memory, which is relatively small and low latency, but shared only by the threads in the same block. Recent GPU models have a L1/L2 cache hierarchy which improves the performance of spatially and temporally local accesses among threads. While the specifics of the GPU design change from one model to the next, the hierarchy of memory resources and the limited ability to communicate are likely to be common to most future SIMT architectures.

The performance of functions executed on the GPU, or *kernels*, can be limited by either the memory bandwidth between the processor and device memory, or the rate at which arithmetic instructions are retired. In most molecular dynamics applications, the bottleneck is the device memory bandwidth. Optimal performance is obtained in these cases by carefully minimizing the amount of memory accessed and by tuning

the access pattern to maximize cache hits.

While the device memory bandwidth is fast, transfers between host memory (accessible by the CPU) and device memory are typically between two and six gigabytes per second, depending on the hardware configuration. Thus, in order to maximize overall application speed, transfers between the host and device must be avoided whenever possible.

Many applications of MD, such as soft matter self-assembly[6,51,66] and protein folding[67–69], necessitate running hundreds of millions of time steps per run and thousands of individual runs. Accelerating the rate at which time steps are performed reduces the time to discovery and enables better predictions through the use of higher fidelity models.

The CUDA C programming environment, which was the first to enable truly general purpose computing on massively parallel GPUs, was released in 2007. GPU-accelerated MD methods were developed shortly thereafter[70–72]. GPU-accelerated implementations of molecular dynamics (MD) have proven to be very fast compared to running a simulation on a single CPU core, achieving two orders of magnitude speed-ups[70,73]. There is a great interest in expanding the algorithms and methods that can be accelerated by GPUs. However, reformulating methods developed for a serial environment, or even a parallel computing environment where work is distributed over many CPU cores, to a massively parallel SIMT environment is not always straightfoward. If a method cannot be effectively implemented in a multi-threaded environment, because of either communication requirements or a stubbornly serial step, the GPU speedup can be lost.

2.5 HOOMD-Blue

HOOMD-Blue[73] is a GPU accelerated Molecular Dynamics Code package originally developed as HOOMD[70] at Ames Laboratory and Iowa State University by

Dr. Joshua Anderson, and now being primarily developed within the Glotzer research group at the University of Michigan. It is available under an open source license[73] and implements the standard algorithms employed by classical MD frameworks. HOOMD-blue differs from most other GPU-accelerated MD methods in that it implements every step of the computation on the GPU and avoids all host/device transfers, except when needed for disk I/O. By avoiding both serial code bottlenecks and slow memory transfers between the host and device, HOOMD-blue reaches maximum performance on a single GPU. In typical benchmarks of Lennard-Jones particle fluids, HOOMD-blue on a current generation GPU is about 80-100 times faster than on a single CPU core.

In each MD time step, the state of system is updated in $\Theta(N)$ time in a number of phases. First, the particles are (1) binned into a cell list. From this cell list (2) a neighbor list is constructed for each particle that contains the indices of all the particles within the specified interaction range. The neighbor list is consulted when (3) computing the pair forces between all interacting pairs of particles. Finally, (4) the computed forces are used to update the particles forward to the next time step. Each phase (1–4) consists of one or more kernels that are executed on the GPU, and all necessary data structures are stored in device memory[70,73]. Different versions of each phase can be interchanged to implement numerous force fields and ensembles, thereby enabling diverse simulation possibilities in a single code package.

2.6 Other Software Resources

Rapid and reliable research depends upon using the computational tools developed by others. In this section, we acknowledge and credit the software tools developed with or by others to do this work.

1. *Building Block Builder (BBB)* BBB is a python interface to HOOMD-blue that allows rapid design of complicated building blocks from sub-building blocks and initialization of a simulation. This software was primarily authored by Joshua Anderson, with contributions from the author as well as other Glotzer Group members.

2. *Freud* Freud is a python interface to HOOMD-blue that allows easy access to and analysis of simulation data. This software was primarily authored by Joshua Anderson, with contributions from the author, as well as other Glotzer Group members.

3. *Large-scale Atomic/Molecular Massively Parallel Simulator, (LAMMPS)* LAMMPS is a classical molecular dynamics code distributed by Sandia National Laboraties[31] which runs on single processors or in parallel. The Glotzer group wrote and contributed several modules for this code, including the fix/rigid nve and nvt modules and the fix ttm module.

4. *Packmol* Packmol[74] is a software that generates initial configurations for MD simulation by packing building blocks into a defined regions of space with flexible constraints. It is developed by Leandro Martnez, State University of Campinas, Brazil.

5. *Visual Molecular Dynamics (VMD)* VMD[75] is a molecular visualization program for displaying, animating, and analyzing large biomolecular (or soft matter) systems using 3-D graphics and built-in scripting. All system snapshots in this dissertation were generated using VMD and rendered with Tachyon[76] or Povray. VMD is developed with NIH support by the Theoretical and Computational Biophysics group at the Beckman Institute, University of Illinois at Urbana-Champaign.

6. *Vorlume* - Vorlume is a computational geometry library developed by F. Cazals, The Algorithms Biology Structure team, INRIA Sophia-Antipolis. Vorlume computes the volume of the domain occupied by the union of balls, as well as the area of the surface bounding the domain.

7. *Voro++* Voro++ is an open source software library for the computation of the Voronoi diagram authored by Dr. Chris Rycroft of UC Berkeley and Lawrence Berkeley Laboratory with contributions from this author.

8. *MedialAxisGenerator* MedialAxisGenerator is a matlab file provided by Suresh Krishnan, University of Wisconsin, that generates the medial axis structure of simple polygons.

Most analysis was done using licensed software packages such as Matlab or Mathematica.

2.7 Computational Resources

Simulations used locally authored code and the GPU-based HOOMD-Blue code package under development in our group, which permitted rapid exploration via simulation. The latter simulations were run on our GPU cluster at the University of Michigan and on the 32-node GPU cluster, AC, at the National Center for Supercomputing Applications on NVIDIA Tesla S1070s. The former simulations using our CPU-based code were run on 2.0 Ghz G5 nodes at the University of Michigan and 2.2 Ghz Opteron Nodes (Jacquard Cluster at National Energy Research Scientific Computing Center).

CHAPTER 3

Rigid Body Calculations on the GPU

The results of this chapter were published in:

Nguyen, Phillips, Anderson, Glotzer, Rigid body constraints realized in massively-parallel molecular dynamics on graphics processing units, Computer Physics Communications, 182 (11), pp 2307-2313, November 2011

Nguyen, Trung, "Computer-aided design of nanostructures from self-and directed-assembly of soft matter building blocks," Dissertation, 2011.

Molecular dynamics (MD) methods compute the trajectory of a system of point particles in response to a potential function by numerically integrating Newton's equations of motion. Extending these basic methods with rigid body constraints enables composite particles with complex shapes such as anisotropic nanoparticles, grains, molecules, and rigid proteins to be modelled. Rigid body constraints are added to the GPU-accelerated MD package, HOOMD-blue, version 0.10.0. The software can now simulate systems of particles, rigid bodies, or mixed systems in microcanonical (NVE), canonical (NVT), and isothermal-isobaric (NPT) ensembles. It can also apply the FIRE energy minimization technique to these systems. In this paper, we detail the massively parallel scheme that implements these algorithms and discuss how our

Algorithm 1 Update bodies, step 1

Require: $\lceil N_{\text{bodies}}/\text{blockDim} \rceil$ blocks are run on the device

$b \leftarrow \text{blockIdx} \cdot \text{blockDim} + \text{threadIdx}$
if $b \leq N_b$ **then**
    $M_b \Rightarrow M$
    $\mathbf{I}_b \Rightarrow \mathbf{I}$
    $\vec{R}_b \Rightarrow \vec{R}_{\text{old}}$
    $\vec{V}_b \Rightarrow \vec{V}_{\text{old}}$
    $\vec{L}_b \Rightarrow \vec{L}_{\text{old}}$
    $q_b \Rightarrow q_{\text{old}}$
    $\vec{F}_b \Rightarrow \vec{F}$
    $\vec{\tau}_b \Rightarrow \vec{\tau}$
    $\vec{V} \leftarrow \vec{V}_{\text{old}} + \frac{\Delta t}{2M} \cdot \vec{F}$
    $\vec{V}_b \leftarrow \vec{V}$
    $\vec{R}_b \Leftarrow \vec{R}_{\text{old}} + \Delta t \cdot \vec{V}$
    $\vec{L} \leftarrow \vec{L}_{\text{old}} + \Delta t/2 \cdot \vec{\tau}$
    $\vec{L}_b \Leftarrow \vec{L}$
    $q_b \Leftarrow \text{Q}(q_{\text{old}}, \Delta t, \vec{L}, \mathbf{I})$
end if

design is tuned for the maximum possible performance. Two different case studies are included to demonstrate the performance attained, patchy spheres and tethered nanorods. In typical cases, HOOMD-blue on a single GTX 480 executes 2.5–3.6 times faster than LAMMPS executing the same simulation on any number of CPU cores in parallel. Simulations with rigid bodies may now be run with larger systems and for longer time scales on a single workstation than was previously even possible on large clusters.

3.1 Implementation

Augmenting HOOMD-blue to include rigid body constraints is accomplished in two parts. First, the following data structures are added to hold the dynamic, static, and computed properties for each body: $\vec{R}_b$, $\vec{V}_b$, q_b, $\vec{L}_b$, M_b, $\mathbf{I}_b$, N_b, B_{bk}, $\vec{D}_{bk}$, $\vec{F}_b$, and $\vec{\tau}_b$. Each quantity with a single subscript is stored in a simple array. Those with two subscripts are stored in rectangular matrices where the second index is the

Algorithm 2 Update particles

Require: N_{body} blocks are run on the device
Require: N I, $\vec{R}$, q, $\vec{V}$, and $\vec{\omega}$ are stored in shared memory.
1: $b \leftarrow$ blockIdx
2: **if** threadIdx == 0 **then**
3: $N_b \Rightarrow N$
4: $\mathbf{I}_b \Rightarrow \mathbf{I}$
5: $\vec{R}_b \Rightarrow \vec{R}$
6: $q_b \Rightarrow q$
7: $\vec{V}_b \Rightarrow \vec{V}$
8: $\vec{L}_b \Rightarrow \vec{L}$
9: $\vec{\omega} \leftarrow \mathbf{R}^T(q)\mathbf{I}^{-1}\mathbf{R}(q)\vec{L}$
10: **end if**
11: syncthreads()
12: **for** $w = 1$ to $\lceil N/\text{blockDim} \rceil$ **do**
13: $k \leftarrow w * \text{blockDim} + \text{threadIdx}$
14: **if** $k \leq N$ **then**
15: $B_{bk} \Rightarrow i$
16: $\vec{D}_{bk} \Rightarrow \vec{D}$
17: $\vec{r}_i \Leftarrow \vec{R} + \mathbf{R}(q) \cdot \vec{D}$
18: $\vec{v}_i \Leftarrow \vec{V} + \vec{\omega} \times (\mathbf{R}(q) \cdot \vec{D})$
19: **end if**
20: **end for**

fastest varying index. Dimensions are sized to the largest body and the leftover space padded with zeroes. Second, new routines are written that integrate the equations of motion of the rigid bodies in the system, with separate versions for the NVE, NVT, and NPT ensembles.

To optimize performance, all data structures are stored in device memory and all integration steps are carried out on the GPU. No communication is required between the host and the device to advance the system. Although padded matrices waste some memory in systems where different bodies contain different numbers of particles, they enable contiguous memory accesses in the integration kernels.

3.1.1 NVE integration kernels

In HOOMD-blue, the integration of Newton's equations of motion for rigid bodies, equations 2.17–2.22, is distributed over five kernels. The first two kernels update the state of the body and its constituent particles. Next, one kernel sums the force and torque on each body from the forces applied to its particles. Finally, two kernels apply the second half of the update to the state of the body and its particles.

Pseudocode describing the basic structure of these kernels is provided in Algorithm 1 and 2. Within the pseudocode, device memory reads/writes are indicated by a double arrow $\Rightarrow/\Leftarrow$ and local memory writes by a single arrow $\leftarrow$. The performance of each of these kernels is bound by device memory bandwidth. Memory accesses are ordered to be contiguous so as to best utilize the cache hierarchy on the GF100 and maximize their performance.

The first kernel, detailed in Algorithm 1, updates the state of the rigid body at the beginning of the time step. Each thread loads state data for its assigned body from global memory, updates the position, orientation, velocity, and angular momentum following Equations 2.17–2.20, and writes the updated state back to global memory. All memory transactions made by Algorithm 1 are contiguous.

The second kernel, detailed in Algorithm 2, sets the constrained position and velocity of each particle that belongs to a rigid body. One block of threads is assigned to each body. At the beginning of the kernel, one thread loads the state of the rigid body into shared memory and a barrier synchronization is performed. Then, all threads participate in computing $\vec{r}_i$ and $\vec{v}_i$. Each thread computes these quantities for several particles i, where $i = B_{bk}$ and $k = \text{threadIdx} + w \cdot \text{blockDim}$, in a loop over $w = 0, 1, 2, 3, ...$ This sliding window construction handles bodies of arbitrary size with a single fixed block size. The matrices B_{bk} and $\vec{D}_{bk}$ are stored with k as their fast index so that the reads on lines 15 and 16 of Algorithm 2 by neighboring threads are contiguous in memory. The writes on lines 17 and 18 may or may not be

contiguous, depending on the order in which particle indices are stored in B_{bk}. To avoid this potential performance hit, all particles in body b are grouped together and listed in order in B_{bk}.

Next, particle-particle forces are computed via the standard MD force calculation kernels. Then the net force $\vec{F}_b$ and torque $\vec{\tau}_b$ on each body are calculated in the third kernel. As in Algorithm 2, one block of threads is assigned to each body. Each thread i loads B_{bk}, $\vec{D}_{bk}$, and the force $\vec{f}_k$ from global memory and the net force and torque are summed using a standard parallel reduction performed in shared memory. The resultant $\vec{F}_b$ and $\vec{\tau}_b$ are then written out to global memory.

In the fourth kernel, the velocity and angular momentum of each rigid body are updated again via equations 2.21 and 2.22. One thread is assigned to each rigid body in a manner analogous to Algorithm 1.

Finally, in the fifth kernel, each body's particles are set to their updated constrained velocity. One block of threads is assigned per body. The kernel is nearly identical to Algorithm 2, except that only the particle velocity is calculated and written to global memory.

All particles that are not part of a rigid body are updated to the next step by the existing standard MD integration kernels. Validation and performance results for these rigid body integration algorithms are provided in Section 3.2.

3.1.2 FIRE energy minimization

The FIRE algorithm[77] works in conjunction with any MD integrator to compute a trajectory to a local energy minimum. At each iteration step, the integrator is used to advance the positions and velocities for all the particles in the system, given the computed forces. FIRE modifies velocities and the step size by the following prescription. As long as the particles in the system are moving in directions that lower the energy of the system as a whole, and have been for a sufficient number of

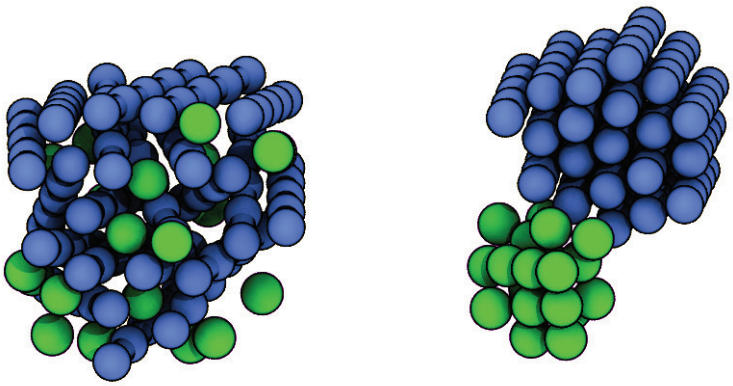


Figure 3.1: (left) Initial configuration of randomly placed rods (blue) intermixed with free particles (green). Rods are attracted to rods and free particles are attracted to free particles. (right) Final configuration after the FIRE energy minimization converges. This image was originally published in reference[1].

steps, particle velocities and the step size are increased, subject to limits. As soon as the particles are no longer moving so as to lower the energy of the whole system, all particles are brought to a halt, the step size is decreased, and new velocities are generated in the direction of the force on each particle. Convergence to a minimum energy is attained when the root mean square force and change in the energy of the system are below set tolerances. In reference[77], FIRE is demonstrated to be effective and surprisingly fast compared to competing schemes, even for systems with millions of degrees of freedom.

We extend FIRE to a system containing rigid bodies by adding the orientation of the rigid bodies to the degrees of freedom and use the rigid body NVE integrator to advance the positions, velocities, orientations, and angular velocities of the bodies. Both the center of mass velocities and the angular velocities of all the bodies are reset to zero if the energy of the system stops decreasing. Convergence is reached when the root mean square force, root mean square torque, and change in the energy of the system are below set tolerances. Reference[77] points out that all degrees of freedom must be comparable for the algorithm to work. In practice, we find that the orientation is a sufficiently comparable degree of freedom and that it does not require

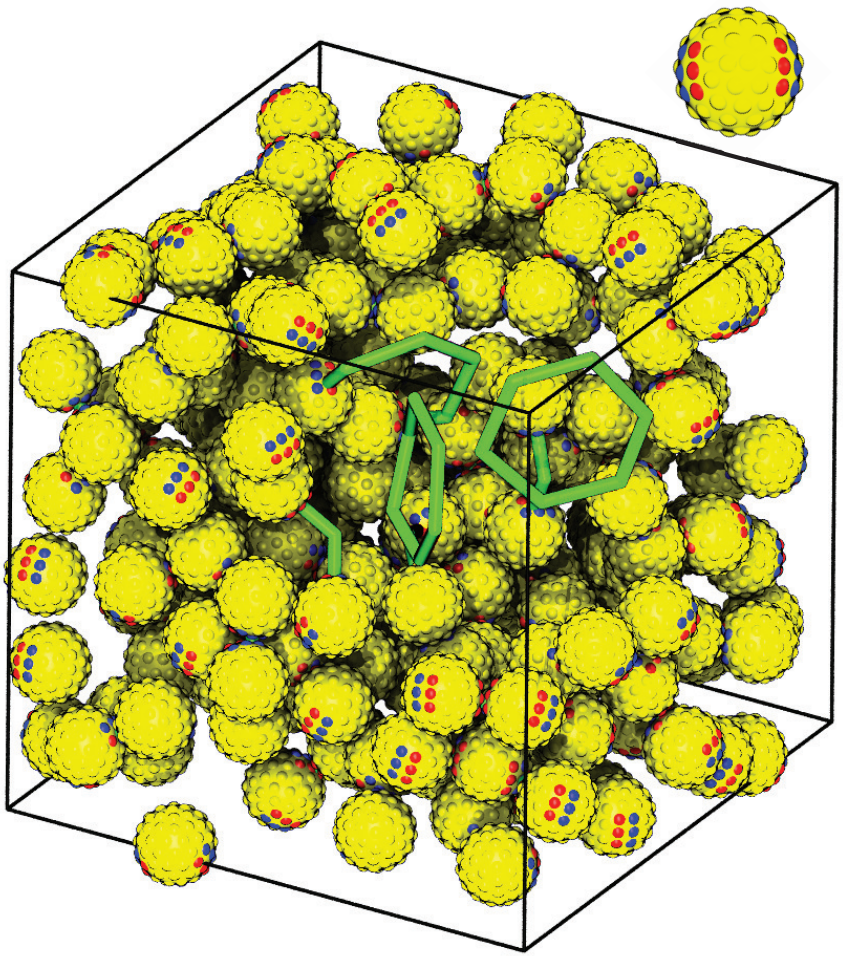


Figure 3.2: A system of 225 patchy spheres, each composed of 90 particles. The red and blue particles are attractive patches on the surface of the body. A single patchy sphere is shown in the upper right for reference. As shown by Zhang[2], these bodies self assemble into rings of six spheres. The spheres have been made invisible in the frontmost octant so that the ring structure formed by the invisible spheres can be shown in green. This image was originally published in reference[1].

special handling.

Figure 3.1 demonstrates FIRE applied to an arrangement of rods and free particles. The rods are rigid bodies composed of five particles arranged linearly. Rod particles interact with other rod particles by the attractive Lennard- Jones (LJ) potential. Free particles also interact by the attractive LJ potential as well. Rod particles and free particles interact by a WCA volume excluding potential. An energy minimization is performed with a force tolerance of 1e-4, a torque tolerance of 0.1, and a change in

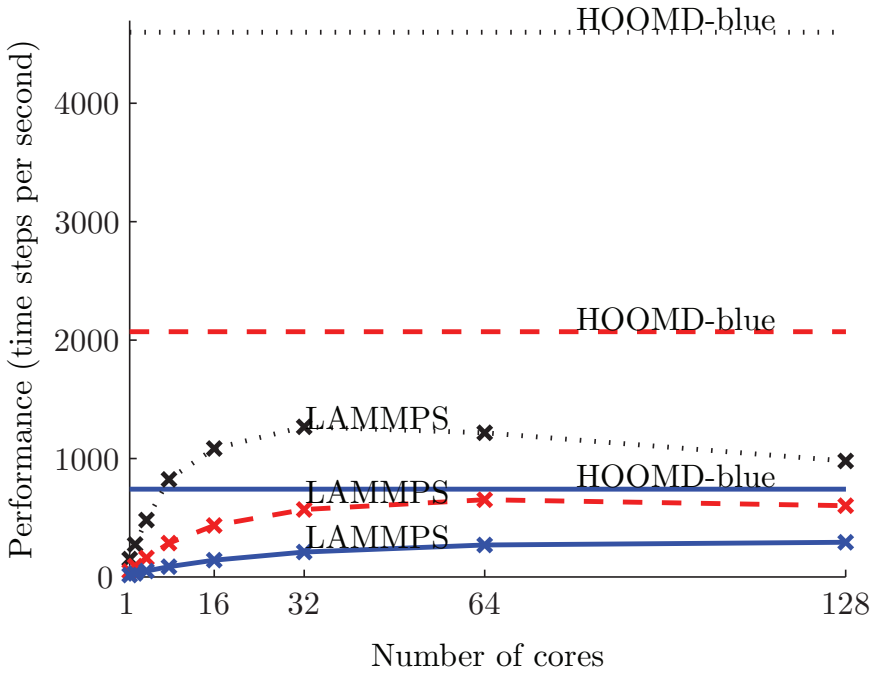


Figure 3.3: Performance in time steps per second obtained while running a simulation of 225 (dotted lines), 667 (dashed lines), and 2000 (solid lines) rigid bodies consisting of 20250, 60030, and 180000 particles respectively. LAMMPS performance on 1, 2, 4, 8, 16, 32, 64, and 128 CPU cores is compared to HOOMD-blue performance on a single NVIDIA GTX 480 (indicated by the horizontal lines). This image was originally published in reference[1].

energy tolerance of 1e-12. The FIRE energy minimization causes the rods to collapse into a hexagonally packed bundle and the free particles to collect into a droplet outside of the rod bundle after 60,684 iterations. Only the first five percent of the time steps are spent collapsing the rod bundle. The rest are needed to collect the dispersed LJ droplets into a single droplet.

3.2 Validation and Performance

The rigid body constraint algorithm is well established in serial and parallel CPU codes[49,60] and is mathematically no different when implemented on the GPU. However, to verify the correct function of our code, various quantities are checked for validity including energy and momentum conservation in the NVE ensemble, as well

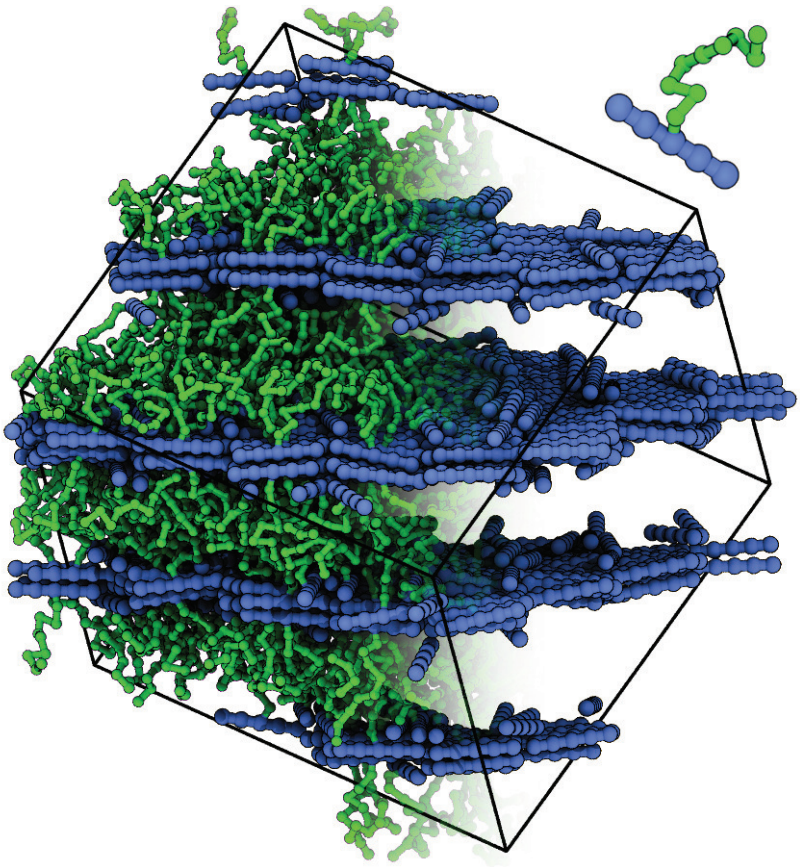


Figure 3.4: A system of one thousand tethered nanorods that have self-assembled into a lamellar bilayer. The upper right inset depicts a single tethered nanorod for reference. Each tethered nanorod is modeled by five particles rigidly connected in a line, attached to a flexible tether of nine particles. Bonds, both rigidly constrained and unconstrained, are shown as cylinders. Tethers have been removed from view in the right half of the image. This image was originally published in reference[1].

as temperature and pressure stability and the correct distribution of energy over the degrees of freedom in the NVT and NPT ensembles. Numerous rigid body systems are also simulated side-by-side on both the CPU and GPU to compare the results and evaluate their relative performance.

The performance scaling of the GPU-accelerated algorithm is tested with simulations of a system of "patchy particles" studied by Zhang et al.[2]. These rigid bodies shall be subsequently referred to as "patchy spheres" to avoid confusion with our usage of the word "particle" referring to the smallest simulation unit. Each patchy sphere is a rigid body composed of 90 particles distributed on the surface of a sphere.

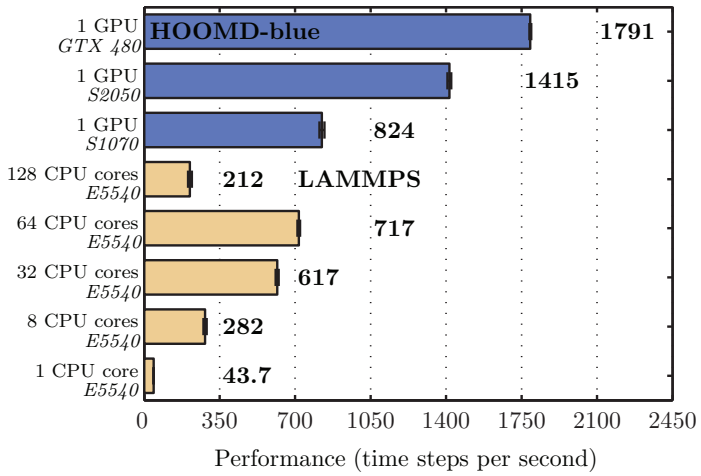


Figure 3.5: Performance in time steps per second obtained while running a simulation of one thousand tethered nanorods (14000 total particles) on various hardware configurations. Each benchmark is performed 50 times. Bars are plotted at the median value and error bars display one standard deviation of variability. This image was originally published in reference[1].

Two attractive patches, each constructed from two linear arrangements of particles that interact with Lennard-Jones pair potentials, are placed at an angle $\theta = 2\pi/5$ with respect to the center of the body. Per Zhang et al.[2], this system self- assembles into rings containing six patchy spheres. The chosen benchmark systems consist of 225, 667, and 2000 patchy spheres resulting in 20250, 60030, and 180000 individual particles, respectively. Each system was annealed to an equilibrium structure at $k_{\mathrm{B}}T = 1.0\varepsilon$. Figure 3.2 shows the system of 225 patchy spheres.

Each simulation is executed using both the LAMMPS and HOOMD-blue code packages. LAMMPS simulations are deployed in parallel over 1, 2, 4, 8, 16, 32, 64, and 128 cores on the Nyx cluster at the University of Michigan. The nodes used are HP ProLiant DL1000 models with Intel® Xeon® e5540 processors operating at 2.53 GHz and connected via 20Gb/s Infiniband. All nodes have identical software configurations, running an x86_64 install of RHEL 5.5, CUDA 3.0, and NVIDIA drivers 195.36.24. The HOOMD-blue simulations were performed on a custom built workstation with a single NVIDIA GTX 480. It also contains an AMD Athlon™ II

X4 630 processor operating at 2.8 GHz and runs CentOS 5.5, CUDA 3.0, and NVIDIA drivers 260.19.21.

Performance results are measured by the number time steps that are executed per second and are shown in Figure 3.3. For the 20K and 60K particle systems, LAMMPS achieves peak performance at 32 and 64 cores, respectively. For the 180K particle system LAMMPS no longer scales well at 128 cores; the performance is only 11% faster than it is at 64 cores. The reason for the poor scaling is the inter-node communication of the rigid body data structures during the time step. LAMMPS uses spatial decomposition to parallelizes a MD simulation over many cores. In simulations of rigid bodies on a CPU cluster, the particles of a given body can be distributed over an arbitrary number of cores. The force and torque summation is performed in LAMMPS by an all-reduce operation that returns results from all nodes to each node[60]. In comparison, the GPU-accelerated implementation is deployed on a single GPU and requires no inter-node or even host-device communication. The equivalent operation to the all-reduce operation is performed within a block on a single streaming multiprocessor. Consequently, HOOMD-blue attains a level of performance for rigid body simulations that cannot be reached with a parallel CPU-only code. For these patchy sphere benchmarks in particular, over a wide range of system sizes HOOMD-blue is 2.5–3.6× faster than LAMMPS at its peak performance for any number of cores.

We also tested systems that mix rigid bodies and unconstrained particles. One example, shown in Figure 3.4, is a system of polymer-tethered nanorods originally studied in reference[51] using LAMMPS. In this simulation, each tethered rod is composed of a five particle rigid rod and a nine particle flexible tether. One thousand tethered rods, for a total of 14000 particles, are placed in a box with packing fraction of 0.22. Rod particles are attracted to each other via a shifted Lennard-Jones pair potential with an interaction cutoff of 2.5 distance units. All other particle interac-

tions are WCA volume excluding. The system is in an NVT ensemble with a kinetic temperature of $k_{\mathrm{B}}T = 1.4\varepsilon$. At these parameters the tethered nanorods self assemble into lamellar bilayers[51].

Simulations are executed with HOOMD-blue on three modern NVIDIA GPUs, a GTX 480, a Tesla S1070, and a Tesla S2050. The Tesla S1070 and S2050 are installed in the Nyx cluster environment where they are hosted by IBM System x3455 nodes each with two AMD Opteron™ 2356 processors operating at 2.3 GHz. The LAMMPS simulations were deployed over 1, 8, 32, 64, and 128 cores of the Nyx cluster in the same configuration used for the patchy sphere runs.

The results of this side-to-side comparison is shown in Figure 3.5. HOOMD-blue running on a GTX 480 executes the tethered nanorod simulation at 1791 time steps per second, which is 2.5× faster than LAMMPS running at peak performance in parallel on 64 CPU cores.

3.3 Conclusion

This chapter discusses how a rigid body constraint algorithm is incorporated into HOOMD-blue, a massively parallel GPU-accelerated MD application. All data structures are stored on the GPU in order to attain the highest level of performance possible by avoiding costly transfers between the host and device. The performance of the kernels implementing the rigid body integration steps is limited only by the device memory bandwidth. This is achieved by carefully avoiding unnecessary device memory accesses and arranging the access patterns so as to make optimal use of the cache hierarchy on the GF100.

Methods for simulating NVE, NVT, and NPT ensembles of rigid bodies are implemented in HOOMD-blue version 0.10.0, which is available free and open source[73]. While two orders of magnitude increases in computational speed over a single CPU core have already been documented for this code package running basic MD simu-

lations[70,73], the GPU is especially well-suited for rigid body constraints. Two case studies are presented in this chapter where HOOMD-blue consistently executes a factor of 2.5–3.6 times faster than the peak performance of the LAMMPS code package parallelized over any number of cores.

This chapter also introduces a modest adaptation to the FIRE energy minimization algorithm that makes it suitable for use with rigid bodies. To our knowledge, HOOMD-blue is the first MD code to allow energy minimization to be applied to systems with rigid body constraints.

With GPU acceleration, MD simulations of systems of rigid bodies can now be carried out on larger systems and for longer time scales on a single workstation than was previously possible even on large clusters. This advance will allow simulations of diverse systems, from molecules and proteins to nanoparticles and colloids, and explorations of previously inaccessible phase spaces.

CHAPTER 4

Massively Parallel Pseudo Random Number Generation for Brownian Dynamics and Dissipative Particle Dynamics

The results of this chapter were published in:
Phillips, Anderson, Glotzer, Pseudo-random number generation for Brownian Dynamics and Dissipative Particle Dynamics simulations on GPU devices, Journal of Computational Physics, 230 (19), 7191-7201, August 2011

Two extensions of the molecular dynamics algorithm are Brownian Dynamics (BD)[41], also known as Langevin Dynamics, and Dissipative Particle Dynamics (DPD)[78,79]. BD and DPD are implicit solvent methods commonly used in models of soft matter and biomolecular systems[40]. BD and DPD enable longer simulation time-scales and larger systems to be studied by abstracting the interaction between the bath of solvent molecules and the larger particles of interest. Simulating the ballistic energy of the numerous solvent particles is replaced by a randomized coarse-grained force controlled by system temperature. Functionally, the interaction acts as a thermostat. BD and DPD are used to model polymers, proteins, nanoparticles, and colloidal systems[80–84].

BD and DPD both require random numbers to be generated at a rate of $\sim kp$ per time step, where p is the number of particles in the system and k is a constant. In a serial environment, these random numbers are typically drawn from a single stream of random numbers generated by a pseudo-random number generator (PRNG). Ignoring all issues of how the numbers generated would be delivered to or from the host CPU

and GPU, using the test hardware of this paper, the GPU is capable of generating more than 140 times as many random numbers per second as the CPU. In general, it is the most efficient for random numbers to be generated as close in the hardware to their intended usage as possible. In a SIMT parallel computing environment where each particle calculation is assigned to a single short-lived thread, small batches of random numbers must be distributed over thousands of threads and millions of kernel calls.

In this chapter, we introduce a novel scheme for generating such widely distributed, small batches of random numbers and then show how this scheme supports BD and DPD simulations. Our scheme, henceforth referred to as one-PRNG-per-kernel-call-per-thread ($pK-pT$) uses a disposable PRNG to produce a micro-stream of random numbers in each thread. The advantage of our scheme over other GPU PRNG schemes is that it does not use the GPU global memory and can accommodate a wide range of PRNG numerical algorithms. We implement our scheme with the *Saru* PRNG package[85] and the Tiny Encryption Algorithm, TEA[86,87]. Given statistically robust sub-algorithms, our scheme is statistically robust, is moderately faster than other schemes for the BD thermostat, and enables a significantly faster algorithm for the DPD thermostat. The $pK-pT$ scheme is currently used to implement BD and DPD in the HOOMD-blue GPU-accelerated MD code package[73].

In Section 4.1, we discuss prior work implementing PRNGs in parallel environments and introduce our $pK-pT$ scheme. In Section 4.2, we briefly introduce how HOOMD-blue accelerates MD, the BD and DPD methods, and how each method uses PRNG schemes. In Section 4.3, we introduce a particular implementation of the $pK-pT$ scheme based on the *Saru* PRNG and TEA, address how we validated our $pK-pT$ scheme, and discuss two micro-benchmarks that we used to measure the performance of the $pK-pT$ scheme against a one-PRNG-per-thread scheme. In Section 4.4, we provide concluding remarks.

4.1 Pseudo-Random Number Generation

Generating robust pseudo-random number streams on a CPU is a well-studied topic[88–90]. In general, most algorithm design choices involve making a trade-off between the statistical robustness of the random number stream and the computational cost of generating the stream. Most trade-off discussions focus on the generation of the PRNG stream without much consideration of how the stream is initialized. With the advent of massively parallel architectures, however, new questions arise regarding how to handle random number generation and both of these considerations are important.

4.1.1 Parallel processor PRNG schemes

In prior work in multi-processor parallel computing environments, a commonly used scheme is the one-PRNG-per-processor scheme. In this scheme, each processor maintains a uniquely seeded and therefore independent random number stream. For example, for MD methods that require random numbers, the Large-scale Atomic/Molecular Massively Parallel Simulator (LAMMPS) code package[31] uses a Marsaglia PRNG[91] with a unique seed generated for each processor. Alternatively, Matsumoto and Nishimura[92] proposed that rather than create independent seeds for each instance of their Mersenne Twister PRNG, instead the processor id could be encoded into the characteristic polynomial used to generate the random number stream.

Another common scheme has been one-PRNG-for-all-processors[93]. Here, a leap-frogging or blocking method is used to partition a single random number stream over many processors. This PRNG scheme is limited to using sufficiently robust PRNGs that also have an efficient method for advancing the state.

In a GPU SIMT environment, these methods, respectively, become one-PRNG-per-*thread* and one-PRNG-for-all-*threads*. Multiple approaches have been considered for the generation of random numbers in different types of applications[87,94–97]. Ref-

erence[94] focuses on either generating a bank of random numbers written to global memory or for use with long calculations executed over a single long kernel call. Langdon[95] proposes using a Parker-Miller PRNG, seeding each thread with a master seed plus the thread number, and discarding the first three random numbers generated. For a BD application, Zhmurov et al.[96] proposes using a separate PRNG on the CPU to generate a sequence of random seeds used to initialize a PRNG, such as Hybrid Tau or Ran2, in each thread. The drawback of these methods for BD is that the random stream of numbers is used over many very short kernel calls. The state of the PRNG associated with each thread is loaded from memory at the beginning of the kernel call and stored to memory at the end of the kernel call. Per-thread resources, possibly even global memory, may be exhausted when storing large PRNG state vectors, thus making their use in real applications impractical. Ran2, for example, has a state size of 35 long integers (64-bits) or 280 bytes per thread, mandating either small thread blocks or expensive repeated reads and writes to global memory[96]. The Marsaglia PRNG used by LAMMPS has an even larger state size of 100 doubles and two integers per stream and, thus is also impractical for a one-PRNG-per-thread scheme. Nearly all of the robust serial PRNGs also have huge state sizes, making them also unsuitable for a one-PRNG-per-thread scheme[90].

For one-PRNG-for-all-threads schemes, either the PRNG must be computationally efficient to advance or partition, or a separate kernel must be called to periodically generate random numbers and "bank" them in global memory for later use. As an example of a stream that can be shared amongst the threads, Zhmurov et al.[96] consider the use of a lagged Fibonacci algorithm. For this PRNG, each thread must still load and store state information, albeit storing far fewer total variables than if this PRNG was used in a one-PRNG-per-thread scheme. Banking random numbers requires a data management scheme in the simulation kernel for expending and refreshing the bank. Also, without a parallel implementation of the PRNG, banking random num-

bers in global memory[94] will become a bottleneck in a massively parallel algorithm. And even if pre-generating such a bank had no computational cost, merely loading a random number into a thread takes more time than generating it in the thread from state information.

4.1.2 One-PRNG-per-kernel-call-per-thread scheme, $pK-pT$

In the $pK-pT$ scheme, instead of loading, modifying, and storing the PRNG state in global memory, we propose to *create* a per-thread PRNG state by applying a *hash-based PRNG* to data unique to each thread and kernel call and then applying a *streaming PRNG* to generate a micro-stream of random numbers. At the end of the thread lifetime, the PRNG state can be discarded. In other words, the state of the PRNG is not stored in global memory.

A hash-based PRNG is a function that applies a sequence of integer operations to an input vector so as to generate a single deterministic output that is effectively decorrelated from the input. In order to achieve a sufficient degree of decorrellation, a large number of operations are applied to the input. In comparison, most streaming PRNGs use few integer operations to efficiently generate statistically random streams.

The usual purpose of a hash-based PRNG is for message encryption. A block of data is passed through a cryptographic hash such as the Secure Hash Function, SHA, with variants SHA-0,-1, or -2, the Message Digest Function 5 (MD5), or the Tiny Encryption Algorithm (TEA) and its extension XTEA, to generate encrypted text that cannot be decrypted without knowledge of the key used. The demands on cryptographic hash functions are very similar to the demands of streaming PRNGs, as any measurable patterns in the output makes the encryption vulnerable to attack.

Using a cryptographic hash alone to generate a small set of random numbers in a massively parallel GPU environment was explored in references[97] and[87]. Tzeng and Wei[97] used MD5 as a random number generator on a GPU for graphics appli-

cations and demonstrated that it produces a high quality set of random numbers. Subsequently, Zafar et al.[87] compared the use of TEA and XTEA against MD5 on a GPU as a random number generator and found that TEA and XTEA also produce random number sets of comparable statistical quality with fewer computations. The limitation of using a hash-based PRNG alone is that the number of random numbers that can be generated in the thread is restricted to the size of the algorithm output, unless the computationally-heavy hash-based PRNG is applied multiple times. By using the hash-based PRNG to create the state for a streaming PRNG, an arbitrary length micro-stream of quality random numbers can be produced for reasonable computational effort.

As long as the micro-stream of random numbers for each calculation can be associated with a unique set of integers, a hash-based PRNG can be applied once to the set to create a unique PRNG state for a micro-stream. References[97] and[87], for example, use lattice point coordinates as inputs. For a MD application, an example of a unique set of input integers is (1) a global user chosen seed, (2) an integer that is incremented in each subsequent kernel call (e.g. the time step), and (3) an integer unique to each micro-stream across a single kernel call (e.g. the thread index). Seeds (1), (2), and (3) define each micro-stream uniquely over a set of simulations, sequence of time steps, and sequence of threads. No memory bandwidth is spent accessing these three integers since seeds (1) and (2) are broadcast to all threads, and since seed (3) uses data already available in the thread.

This scheme takes advantage of the massive parallelism of a GPU, whereby the long memory latencies can be hidden by arithmetic instructions performed in other concurrently running threads. An ideal ratio of arithmetic to global memory accesses enables full utilization of both the memory bandwidth and instruction throughput[64]. While the ideal ratio is device dependent, the value for one specific GPU, the NVIDIA Tesla C2050, is 27.8 instructions per 32 bits loaded. In a simulation large enough

that the streaming multiprocessors are sufficiently occupied, a hash-based PRNG that requires 28 instructions to initialize the state vector for a streaming PRNG can be just as computationally efficient as a one-PRNG-per-thread scheme that loads a small 32-bit state.

Also, as the computational throughput in GPU devices is currently increasing faster than the memory bandwidth, trading loads and store for computations is likely to remain a winning strategy. For a kernel that is memory bound without considering the generation of random numbers, the $pK-pT$ scheme will generally outperform any scheme requiring memory accesses.

4.2 SIMT Molecular Dynamics, Brownian Dynamics, and Dissipative Particle Dynamics

In molecular dynamics (MD) algorithms, the trajectory of a system of particles is generated by solving Newton's equations of motion repeatedly for suitably small time steps. HOOMD-blue, which implements all computations on the GPU, realizes a classical MD algorithm by employing a velocity Verlet integration scheme. The calculation and summation of short-range forces between pairs of particles is reduced to an $\Theta(N)$ calculation by using cell lists and neighbor lists. A given particle's neighbor list contains the indices of all particles within interaction range of that particle. The net pair-wise force acting on a particle can be determined by consulting only those particles in that particle's neighbor list.

In HOOMD-blue, the basic kernels deployed over a single time step are (1) a first update of the position and velocity of the particles (2) the calculation and summation of the forces on each particle, and (3) a second update of the velocity of the particles. When needed (typically every 9 time steps), (4) the cell list for the system is recalculated and (5) the neighbor list for each particle is built. All the kernels are handled

by assigning a thread to each particle[70,73].

One significant difference between SIMT MD algorithms and a streaming serial MD algorithm is how the pair-wise force summation is handled. On the CPU, the calculation and summation of pair-wise forces is generally made more efficient by calculating the force only once per particle pair[40]. If particle A and particle B interact, the interaction force can be calculated while looping over the neighbors of particle A, and then the equal and opposite force acting on particle B can be written to a globally stored force array. In contrast, most massively parallel molecular dynamics algorithms that employ one thread per particle have each thread calculate all the forces acting on a given particle[70,98–101], with a the texception of references[102] and[103]. This doubles the number of non-bonded force computations, but converts a gather-scatter memory access pattern to a simple gather memory access pattern[104] leading to an overall faster algorithm.

4.2.1 Brownian dynamics

The BD thermostat models the solvent-colloidal particle interaction by applying a Langevin force[40,41]. In BD, a temperature dependent random force and a drag force proportional to the particle velocity is applied to each particle at each time step of the simulation. The force F applied to each spatial component of particle i is

$$F_i = -\gamma v_i + R_i\sqrt{3}\sqrt{\frac{2k_{\mathrm{B}}T\gamma}{\Delta t}} \tag{4.1}$$

where γ is a coupling constant, T is the temperature of the solvent, Δt is the time step size, v_i is the velocity of the particle, k_{B} is the Boltzmann constant, and R_i is a random number uniformly distributed in the range [-1,1]. To compute this force for a given particle, three random numbers (one for each x, y, and z component) and the velocity of the particle are required. We choose to incorporate the force calculation

into the second update of the velocity of the particles, kernel (3).

The one-PRNG-per-thread, -for-N-threads, and -for-all-thread schemes are all suitable for generating the single micro-stream of three random numbers required per particle per time step. Algorithm 3 provides a description of a BD kernel using a one-PRNG-per-thread scheme.

To use the $pK{-}pT$ scheme for BD, the hash-based PRNG is applied to (1) a global seed, (2) the time step of the simulation, and (3) the particle index (i.e. the thread index) to generate an initialized state vector for the streaming PRNG. The PRNG state data is created and discarded inside the thread, and therefore does not need to access global memory. Algorithm 4 provides a description of a BD kernel using a $pK{-}pT$ scheme.

4.2.2 Dissipative particle dynamics

The net sum of all the Langevin forces applied to a system for the BD thermostat is not zero because the random forces are not computer pairwise, so momentum is not conserved and therefore hydrodynamic behavior is not preserved (without the introduction of interaction tensors[105]). In contrast, the DPD thermostat method models the solvent-colloidal particle or bead interaction by applying the Langevin force to all pairs of interacting particles. Equal and opposite random and drag forces are applied to particle pairs, so the momentum is conserved both locally and system-wide. The force F applied to a pair of particles i and j in DPD, where $r_{ij} < r_{interaction}$ is

$$\begin{aligned} F_{ij} = \quad & F_c(r_{ij}) - \gamma[\omega(r_{ij})]^2(\vec{v}_{ij} \cdot \hat{r}_{ij}) \\ & -R_{ij}\omega(r_{ij})\sqrt{3}\sqrt{\frac{2k_bT\gamma}{\Delta t}} \end{aligned}$$

where F_c is a conservative force, γ is a coupling constant, T is the temperature of the solvent, dt is the time step size, $\vec{v}_{ij}$ is a velocity difference vector $\vec{v}_{ij} = \vec{v}_i - \vec{v}_j$, $\hat{r}_{ij}$ is a unit vector $\hat{r}_{ij} = (\vec{r}_i - \vec{r}_j)/|\vec{r}_i - \vec{r}_j|$, $\omega(r_{ij})$ is a function of the distance between particle i and j, and R_{ij} is a random number uniformly distributed in the range [-1,1]. A single random number is required for each pair-wise applied Langevin force between interacting particles. Depending on the density of the system and the interaction radius, this may require generating ∼50-100 random numbers per particle.

While the DPD algorithm effectively applies a pair-wise force to the system of particles, that force cannot be simply incorporated into the existing pair-wise force summation kernel (2) using a one-PRNG-per-thread, per-N-threads, or for-all-threads scheme. In kernel (2), the same random force must be applied in both threads that compute the ij interaction. This implies that either some model of communication between threads, some method of two threads (and only those two threads) loading the same random number from a bank, or for this algorithm and only this step, a separate fine-grained decomposition based on one-thread-per-force calculation, with all the accompanying data structures, must be implemented. The first option, modifying kernel (2) to include a scatter communication using atomic operations to write to a global force array, is the simplest. This communication significantly slows the kernel execution time, as shown by our later benchmarks. Algorithm 5 provides a description of such a DPD kernel using a one-PRNG-per-thread scheme.

The DPD pair-wise force can be simply incorporated into the existing pair-wise force summation kernel (2) if using the $pK{-}pT$ scheme. The $pK{-}pT$ scheme enables two threads to generate the same random number, and thus the same stochastic force, without requiring coordination between the threads. To use the $pK{-}pT$ scheme for DPD, the hash-based PRNG is applied to (1) a global seed, (2) the time step of the simulation, and (3) the index of the first (4) and second particle in the interaction. This generates an initialized state vector for the streaming PRNG that is the same

in both threads. Algorithm 6 provides a description of a DPD kernel using a $pK{-}pT$ scheme.

Algorithm 3 One-PRNG-per-thread (Brownian Dynamics)

if $tid \leq n_{\text{particles}}$ **then**
  load $RNGstate \leftarrow d\_RNGstate[tid]$
  load $v \leftarrow d\_velocity[tid]$
  load $f \leftarrow d\_force[tid]$
  $R_x \leftarrow \text{callRNG}()$
  $R_y \leftarrow \text{callRNG}()$
  $R_z \leftarrow \text{callRNG}()$
  $f = f + \text{F}(v, R_x, R_y, R_z)$
  $v = \text{update\_velocity}(v, f)$
  store $d\_RNGstate[tid] \leftarrow RNGstate$
  store $d\_velocity[tid] \leftarrow v$
  store $d\_force[tid] \leftarrow f$
end if

Algorithm 4 One-PRNG-per-kernel-call-per-thread (Brownian Dynamics)

Require: timestep, seed are broadcast
if $tid \leq n_{\text{particles}}$ **then**
  load $v \leftarrow d\_velocity[tid]$
  load $f \leftarrow d\_force[tid]$
  $RNGstate \leftarrow \text{hashRNG}(timestep, seed, tid)$
  $R_x \leftarrow \text{callRNG}()$
  $R_y \leftarrow \text{callRNG}()$
  $R_z \leftarrow \text{callRNG}()$
  $f = f + \text{F}(v, R_x, R_y, R_z)$
  $v = \text{update\_velocity}(v, f)$
  store $d\_velocity[tid] \leftarrow v$
  store $d\_force[tid] \leftarrow f$
end if

4.3 Validation

4.3.1 *SaruSaru* and *Saru*TEA PRNG

For performance testing the $pK{-}pT$ scheme we considered two combinations of a hash-based and streaming PRNG. The PRNG package *Saru*[85]contains both a hash-

Algorithm 5 One-PRNG-per-thread (Dissipative Particle Dynamics)

if $tid \leq n_{\text{particles}}$ **then**
  load $v \leftarrow d\_velocity[tid]$
  load $x \leftarrow d\_position[tid]$
  load $nlist\_length \leftarrow d\_nlist\_length[tid]$
  $f_{net} \leftarrow 0$
  load $RNGstate \leftarrow d\_RNGstate[tid]$
  for $i < nlist\_length$ **do**
    load $nid \leftarrow d\_nlist[tid, i]$
    load $v_{neigh} \leftarrow d\_velocity[nid]$
    load $x_{neigh} \leftarrow d\_position[nid]$
    $R \leftarrow \text{callRNG}()$
    $f_{tid,nid} = \text{F}(v, x, v_{neigh}, x_{neigh}, R)$
    $f_{net} \leftarrow f_{net} + f_{tid,nid}$
    atomic $force[nid] \leftarrow force[nid] - f_{tid,nid}$
  end for
  atomic $force[tid] \leftarrow force[tid] + f_{net}$
  store $d\_RNGstate[tid] \leftarrow RNGstate$
end if

Algorithm 6 One-PRNG-per-kernel-call-per-thread (Dissipative Particle Dynamics)

Require: timestep, seed are broadcast
if $tid \leq n_{\text{particles}}$ **then**
  load $v \leftarrow d\_velocity[tid]$
  load $x \leftarrow d\_position[tid]$
  load $f \leftarrow d\_force[tid]$
  load $nlist\_length \leftarrow d\_nlist\_length[tid]$
  for $i < nlist\_length$ **do**
    load $nid \leftarrow d\_nlist[tid, i]$
    load $v_{neigh} \leftarrow d\_velocity[nid]$
    load $x_{neigh} \leftarrow d\_position[nid]$
    $RNGstate \leftarrow \text{hashRNG}(timestep, seed, tid, nid)$
    $R \leftarrow \text{callRNG}()$
    $f_{tid,nid} = \text{F}(v, x, v_{neigh}, x_{neigh}, R)$
    $f \leftarrow f + f_{tid,nid}$
  end for
  store $force[tid] \leftarrow f$
end if

based and streaming PRNG. The streaming PRNG of *Saru* was used in conjunction with its native hash-based PRNGs and with an implementation of TEA with eight rounds (TEA8) as recommended by reference[87]. These two combinations will be referred to as *SaruSaru* and *Saru*TEA respectively.

Streaming PRNGs are tested for statistical randomness by applying a large set of theoretical and practical tests to its output stream. Hash-based PRNG can be tested by applying the same tests to concatenations of their outputs while systematically varying the inputs. TestU01 is a software library offering a collection of utilities for the empirical statistical testing of uniform random number generator[90] (other test batteries include DIEHARD, NIST, Rabbit, and Gorilla). Each independent PRNG component we considered has been tested and found statistically robust.

Saru provides a choice of three hash-based PRNGs, able to transform one, two or three seeds (s_1, s_2, s_3) into a two integer state for its streaming random number generator, using bitwise xor, multiplication, addition, bit shifting, and type conversion, via 12, 30, and 45 issued instructions, respectively. The hash-based PRNG for *Saru* was tested successfully by its author against TestU01's Crush[85,90]. The streaming PRNG in *Saru* has a state of two 32-bit words, and it uses a linear congruential generator (LCG) and an Offset Weyl Sequence (OWS) to advance the two words. *Saru* mixes the words and further transforms the output by xors and a multiply. This streaming PRNG has a period of $3666320093 \cdot 2^{32} \approx 2^{63.77}$ and requires 13 issued instructions to advance the state and generate an output. The streaming PRNG for *Saru* was tested successfully by its author against DIEHARD, Rabbit, Gorilla, and TestU01's SmallCrush, Crush, and BigCrush[85,90].

The Tiny Encryption Algorithm[86], TEA, has a 64-bit input and uses a 128 bit key to generate a 64 bit output. Each round, or unit set of integer and bitwise operations, of TEA uses bitwise xor, multiplication, addition, and bit shifting to mix the input with the key. Each round of TEA involves 17 issued instructions, for a total of 136

issued instructions for TEA8. In reference[87], eight rounds were sufficient to produce high quality random numbers when tested against NIST and DIEHARD. As TEA was designed to minimize memory footprint while maximizing speed, it is particularly ideal for GPU applications.

The hash-based PRNGs are seeded as follow. For the Saru hash-based PRNG, we premix certain inputs so as to reduce three (or four) unique integers to two (or three) integers and then used the two (or three) input hash-based PRNG. For both BD and DPD, the global seed, gs, provided by user is hashed once by $gs = gs \times 0x12345677 + 0x12345;\ gs = gs \wedge (gs >> 16);\ gs = gs \times 0x45679$. This hash is performed as a precaution as users tend to use a very restricted set of seeds. For BD, seed s_1 is set to the particle index. Then we add the time step and the global seed to generate s_2. For DPD, s_1 is set to the smaller particle index, and s_2 is set to the larger particle index, then we again add the hashed global seed and the time step to generate s_3.

For the TEA8 hash-based PRNG, we use the key provided by reference[87], or $\{k_1$ = 0xA341316C, k_2= 0xC8013EA4, k_3= 0xAD90777D,k_4 = 0x7E95761E $\}$. For BD, the two inputs are set to the particle id and the time step. The first part of the key, k_1 is then set to the hashed global seed. For DPD, the two inputs are set to the smaller and larger particle index, in that order, and the first and second part of the key are set to the time step and hashed global seed. Using parts of the key for the extra inputs allows a single application of TEA8 to be used rather than using a nested hash function which requires further iterations of TEA8.

Both the *Saru* hash-based PRNG and TEA8 output two 32-bit words. The state of the *Saru* streaming PRNG is simply set to the output.

Even if the hash-based PRNG and the streaming PRNG are individually statistically robust, we must also validate statistical randomness between multiple streams generated by this scheme. This is effectively a validation of the hash-based seeding of the PRNG, insuring that different, highly correlated seed values do not generate

correlations. This is also a validation that the streaming PRNG does not inadvertently undo the mixing of the hash-based PRNG. It is also important to test PRNG streams in a manner that reflects their use. The PRNG output used by a massively parallel simulation might best be visualized as a matrix. Each column represents the PRNG stream of a single thread over sequential kernel calls. Each row represents the PRNG values generated over a single time step. We want to analyze the randomness of this matrix both across the rows and across the columns. In practice, this matrix of values could be considered not just 2D, but 3D or 4D since the seeding values themselves are multidimensional. PRNGs in parallel environments should be sliced and concatenated into an appropriate single stream for testing. The same tool set can be applied to streams formed by reading through the matrix by different paths.

The two implementations used for the $pK{-}pT$ scheme were tested in three ways. First, we assumed 16,000 particles and generated micro-streams of three random numbers per particle. The micro-streams of all the particles were concatenated and then concatenated again over subsequent time steps in order to test for whole system correlations. Second, we concatenated the micro-streams for a single particle over many time steps to test for correlations in the stochastic force that may be applied to a single particle. Third, to model the DPD stream, we assumed that a single particle was interacting with the same 50 particles for all time, and concatenated those random numbers over time. We applied the TestU01 SmallCrush, Crush, and BigCrush to each stream, generating a net total of 319 test statistics and p-values per stream. All three of these stream types passed the TestU01 SmallCrush, Crush, and BigCrush test batteries, with only spurious non-systemic failures of a few sub-tests for some seed values, a failure rate in the range of the expected number of failures for TestU01.

4.3.2 Benchmarks

To test the performance of the $pK{-}pT$ scheme we used micro-benchmarks that emulate the kernels of the MD code[70,73]. We compared the performance of the $pK{-}pT$ scheme against a one-PRNG-per-thread (pT) scheme, where the small state (two 32-bit unsigned integers) of the *Saru* streaming PRNG is loaded and stored during each kernel call. The pT scheme used represents a best case memory access pattern relative to the per-thread, per-N-threads, or for-all-threads schemes. We also compared the $pK{-}pT$ scheme using *SaruSaru* and *Saru*TEA to a pT scheme using the *XORWOW* PRNG provided in the recently released CURAND library, available in the CUDA™3.2 toolkit. As the current seeding method provided in the CURAND library for *XORWOW* is very slow (requiring ≈ 50 times longer than what is needed to load and store the state), a $pK{-}pT$ scheme with the *XORWOW* PRNG was not considered.

All benchmarks were performed on a custom built workstation with an AMD Athlon™II X4 630 CPU operating at 2.8GHz on a mainboard with the nForce 980a chipset, 4GB of DDR3 RAM operating at 1333MHz, and an NVIDIA® GeForce® GTX 480 operating at stock settings with a processor clock of 1401MHz and a memory clock of 1848MHz. The system runs an x86_64 installation of CentOS 5.5, the CUDA toolkit version 3.2.9 and corresponding GPU device driver 260.24. The GTX 480 is the latest "Fermi" architecture which is the first generation of NVIDIA GPUs to include atomic operations for floats, enabling Algorithm 5.

4.3.2.1 Brownian dynamics

For the BD micro-benchmark kernel, (Algorithm 3 and 4), the current force and velocity vectors for a particle are retrieved from memory, the Langevin force is calculated and added to the force vector, the velocity of the particle is updated, and both the force and velocity vector are written back to memory. This minimal ker-

nel requires 48 bytes of memory transfer in each thread (presuming the force and velocity vector are both three 32-bit floats). Retrieving and storing the state of the *Saru* streaming PRNG and *XORWOW* PRNG requires an additional 16 bytes and 80 bytes, respectively, of memory transfer. The threads are synchronized at the end of each kernel call and the kernel is called a thousand times to average out statistical variation.

The *SaruSaru* and *Saru*TEA $pK{-}pT$ scheme reduces the memory access by 25% relative to the *Saru* pT scheme. In Figure 4.1, the average time spent per kernel invocation is shown for systems of 10,000 to 100,000 particles. The *SaruSaru* and *Saru*TEA$pK{-}pT$ scheme out-performs the pT scheme over the entire range. For systems of 100,000 particles, the *SaruSaru* $pK{-}pT$ scheme is 16% faster than the *Saru* pT scheme. The *Saru*TEA scheme, with its three times larger hash-based PRNG, is only 7% faster than the *Saru* pT scheme. In contrast, the *XORWOW* PRNG in a pT scheme is three times slower than the *Saru* PRNG pT scheme due to loading and storing the larger PRNG state.

4.3.2.2 Dissipative particle dynamics

For the DPD micro-benchmark, the DPD thermostat is applied to each particle as in a MD force summation kernel[70], as described in Algorithm 5 and 6. Each thread loads its particle's neighbor list one entry at at time. The position and velocity of each neighbor is then loaded via texture reads, and the pair-wise Langevin forces are computed. For this benchmark, the conservative force is not computed. The threads are synchronized at the end of each kernel call and the kernel is called a thousand times to average out statistical variation.

For the pT scheme, the current velocity and position is loaded at the beginning of the kernel, and a half-neighbor-list is used. Half-neighbor-lists are neighbor-lists where, if particles i and j are neighbors, particle i is in particle j's neighbor-list or

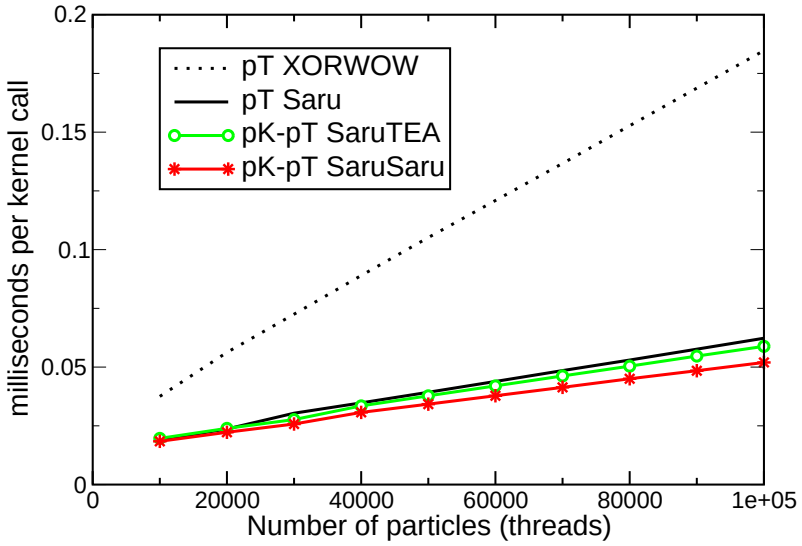


Figure 4.1: For a Brownian Dynamics micro-benchmark, a one-PRNG-per-thread, pT, scheme using the *Saru* and *XORWOW* streaming PRNG, is compared to a one-PRNG-per-kernel-per-thread, $pK{-}pT$, scheme using *SaruSaru* and *SaruTEA* combined PRNGs. This image was originally published in reference[3].

vice-versa, but not both. Floating point atomic adds are used for all writes to the force array in global memory and are bundled load-add-store operations. The PRNG state is loaded at the beginning of the kernel, a single stream is used during the kernel, and the PRNG state is stored at the end of the kernel. The total average memory transfer per thread is 72 bytes + $(N/2)$*60 bytes, where N is the number of neighbors per particle* .

For the $pK{-}pT$ scheme, the current velocity, position, and force of the particle is loaded at the beginning of the kernel call and a full neighbor-list is used for each particle. Full neighbor-lists are neighbor lists where, if particles i and j are neighbors, particle i is in particle j's neighbor list, and vice versa. The PRNG is continually re-seeded for each for each pair as described above. At the end of each kernel call,

*velocities and positions accessed by texture reads and therefore are float4's

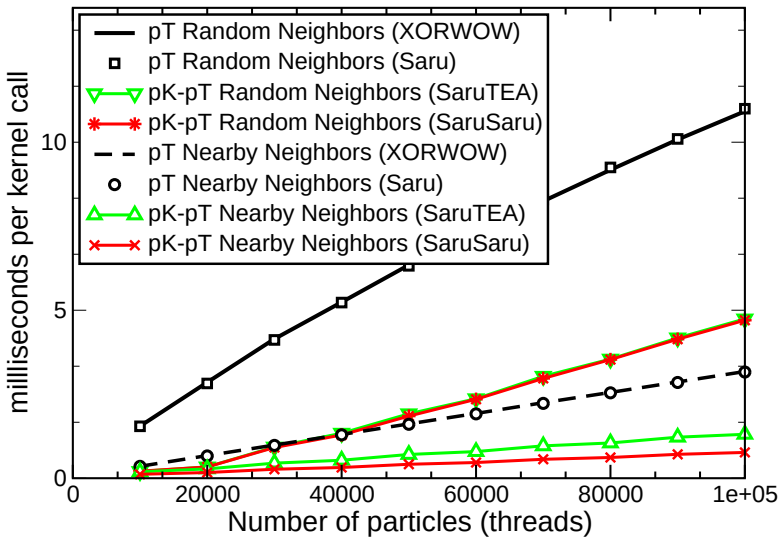


Figure 4.2: For a Dissipative Particle Dynamics micro-benchmark, a One-PRNG-per-thread, pT, scheme using the *Saru* and *XORWOW* streaming PRNG, is compared to a one-PRNG-per-kernel-per-thread, $pK-pT$, scheme using *SaruSaru* and *SaruTEA* combined PRNGs. Two types of neighbor lists, with different topologies and therefore different patterns of memory usage, are shown to bound actual simulation performance. This image was originally published in reference[3].

the net force on the particle is written to global memory. The total memory transfer of each thread is 56 bytes $+N$*36 bytes, where N is the number of neighbors per particle.

In simulations of diffusive particles, after a certain amount of time there is no correlation between the indices of nearby particles. HOOMD-blue solves this by periodically reorganizing particle data so that nearby particles are likely to have nearby indices[70]. Therefore, two topologies of neighbor list were constructed to bound the possible neighbor lists. In one case, particles are assumed to only interact with neighbors with nearby indices $(\pm N/2)$. In the other case, the neighbor list was randomly constructed. In both cases the half-neighbor list was constructed so that each particle is assigned roughly (or for the case of sequential neighbors, exactly)

the same number of neighbor calculations to balance the load of work among threads. The two types of neighbor lists lead to significantly different patterns of cache use and data collisions in global memory accessed by the computation kernel. The number of neighbors, N, was set to 50.

The pT scheme, which uses both a gather and scatter memory access pattern, has 15% fewer memory transfers than the $pK{-}pT$ scheme, which only uses a gather memory access pattern. This is because the pT scheme uses half-neighbor lists. However, a quarter of the memory transfers for the pT scheme are significantly slower atomic operations.

In Figure 4.2, the average time spent per kernel invocation of this micro-benchmark is shown for systems of 10,000 to 100,000 particles. The *SaruSaru* and *Saru*TEA $pK{-}pT$ scheme out-performs the pT scheme over the entire range. For nearby neighbors, the *SaruSaru* $pK{-}pT$ scheme is 3 to 4 times faster than the pT scheme. For randomly dispersed neighbors, the *SaruSaru* $pK{-}pT$ scheme is 2 to 7 times faster than the pT scheme. The *Saru*TEA $pK{-}pT$ is 60-70% slower than the *SaruSaru* $pK{-}pT$ scheme for nearby neighbors, but performs equivalently for randomly dispersed neighbors. In effect, the inefficient memory operations necessary to load data for randomly dispersed particles interleave with arithmetic operations on the GPU multiprocessor so as to completely hide the larger TEA8 hash-based PRNG.

Unlike the BD kernel, the time to load and store the PRNG state is negligible relative to the length of the kernel. Thus, the performance difference of the kernels using the pT scheme and the *Saru* or *XORWOW* PRNG is indistinguishable.

4.4 Conclusion

In this chapter, we introduce a one-PRNG-per-kernel-per-thread scheme, which allows the generation of millions of micro-streams of random numbers in a SIMT massively parallel computing environment without having to load and store a PRNG state

vector in each thread. This scheme is currently used in HOOMD-blue to support BD and DPD simulations. The one-PRNG-per-kernel-per-thread scheme uniquely supports DPD simulations without changing the one-thread-per-particle fine grained parallelism used for the MD algorithm or requiring communication and coordination between threads via atomic operations. By eliminating the need to load and store PRNG state information in the kernel, the one-PRNG-per-kernel-call-per-thread scheme is moderately faster than other schemes for calculating small batches of random numbers in the BD algorithm. By eliminating the need to communicate and coordinate between threads, the one-PRNG-per-kernel-per-thread scheme is 2-7 times faster than a one-PRNG-per-thread scheme for the DPD thermostat.

The one-PRNG-per-kernel-per-thread scheme utilizes both a hash-based and a streaming PRNG in each kernel call. While for our simulation software package we chose the *SaruSaru* PRNG, which has both PRNG functionalities, the components of the PRNG scheme can be changed to meet the needs of the application. Replacing the *Saru* hash-based PRNG with TEA8, for example, is a reasonable alternative that more naturally handles up to six unique inputs. Replacing the streaming PRNG with a one that has a longer period may also be more suitable for some applications. This PRNG scheme also allows for a mix of different types of PRNGs to be used in a massively parallel application if desired.

To our knowledge, HOOMD-blue is the only GPU MD code package that can perform Dissipative Particle Dynamics simulations. An example of a full DPD benchmark simulation comparing GPU and parallelized CPU performance is shown in Figures 4.3 and 4.4. Figure 4.3 shows a block copolymer system of 2400 A_3B_7 polymers (24,000 particles) that have self-assembled into the hexagonally packed cylinder phase[4,5] simulated using the DPD method. In Figure 4.4, we compare the performance, measured in time steps per second, of HOOMD-blue and a optimized parallel multiprocessor (CPU) MD code package LAMMPS[31] in simulating this system. The

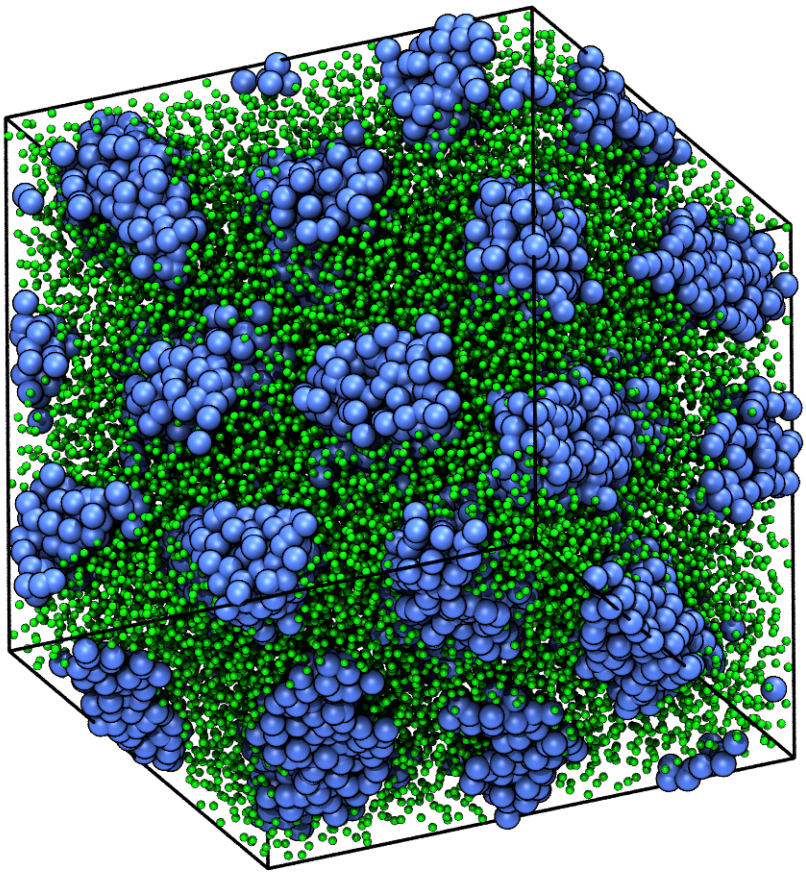


Figure 4.3: Pictured is the full benchmark system for the DPD simulation method, a A_3B_7 block copolymer system of 2400 polymers (24,000 particles) self-assembled into the hexagonally packed cylinder phase. Details on this system can be found in reference[4,5]. This image was originally published in reference[3].

GPU code run on a single NVIDIA GTX 480 is significantly faster than the CPU code parallelized over 64 cores. By enabling BD and DPD to be performed in HOOMD-blue, a broad range of mesoscale coarse-grained simulations can now be accelerated in a massively parallel architecture, thus also accelerating the understanding of soft matter and biomolecular systems and discovery of new materials.

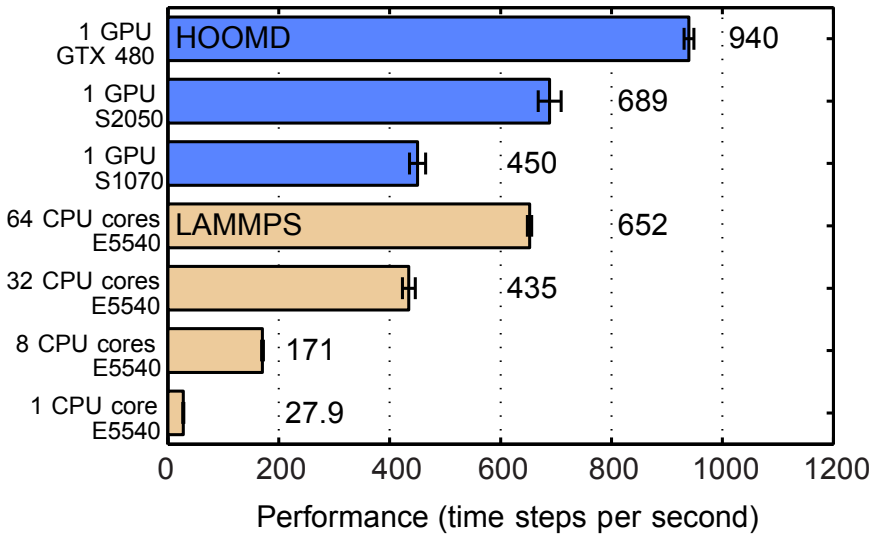


Figure 4.4: Benchmarks comparing HOOMD-blue to LAMMPS, a parallelized CPU molecular dynamics code package, for the DPD Benchmark of Figure 4.3. This image was originally published in reference[3].

CHAPTER 5

Filling - A New Shape Packing and Covering Optimization Problem

The results of this chapter and the next will be published in:
Phillips, Anderson, Huber, Glotzer, The Filling Problem; Combining Packing and Covering to Optimally Fill Shapes, Preprint
Phillips, Anderson, Chen, Glotzer, Optimally Filling Polygons with Discs, Preprint

The properties of packing non-overlapping objects, such as monodisperse or polydisperse, spheres, ellipsoids, or tetrahedra in an optimal, random, flowing, or jammed configuration, has been long-studied by physicists[106–116]. The question of how a lattice or space can be covered is also a fundamental question to many physics problems[117–121].

In this chapter, we present a new problem of spatial subdivision that is a hybrid of the familiar packing and covering problems. We define *filling* as the problem of packing overlapping objects inside of a defined shape such as to optimally cover the interior volume without extending beyond the boundary of the shape (Figure 5.1). We are primarily interested in the optimal filling of an n-dimensional shape with a well-defined $n-1$ surface with n-dimensional polydisperse balls. This question arises in our research in the modeling of anisotropic nanoparticles as a rigid body composed of a sum of isotropic volume-excluding potentials[2,5,30,122,123].

By using a shifted WCA potential and defining a set of points as a rigid body, a filling solution of a shape can be used to model the volume-excluding potential of

"hard" extended objects, including faceted shapes. In Figure 5.2 below, for example, each "pentagon" is actually composed of 31 discs. Using a BD simulation, the pentagons were compressed and formed the same striped phase composed of alternating rows of oppositely pointing particles as shown by Schilling et al[124] using a Monte Carlo simulation of hard pentagons. Using a BD Simulation in Figure 5.3, "tetrahedra" composed of 81 balls and attractive patches on three faces and one vertex have self-assembled into icosahedral clusters.

The filling problem as defined here also has applications to the problem of irradiating a tumor with the fewest number of beam shots, while controlling the beam diameter, but without damaging surrounding tissue[125], using time-delayed sources to create shaped wave fronts, combining precision placed explosives with tunable blast radii, positioning proximity sensors with defined radii, or any problem of ablation or deposition where one has a sharp impenetrable boundary and a radially tunable tool with a cost per use. It also may be related to coarsened (due to Ostwald ripening) wet foams packed in containers with non-wetting surfaces.

In the deceptively simple problem of determining the optimal set of discs to fill an arbitrary planar shape, we find a surprisingly rich problem, with many open questions.

We define *filling* as the problem of packing overlapping objects inside of a defined shape such as to optimally cover the interior volume without extending beyond the boundary of the shape. We are primarily interested in the optimal filling of an n-dimensional shape with a well-defined $n-1$ surface with n-dimensional polydisperse

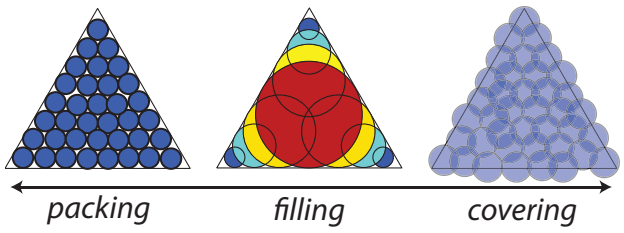


Figure 5.1: The problem of filling a shape, such as a triangle, can be viewed as a combination of a packing and covering problem.

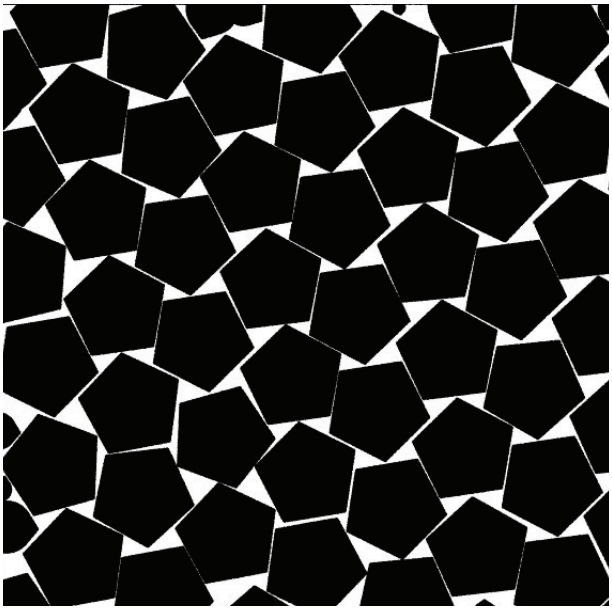


Figure 5.2: Each pentagon is actually composed of 31 discs.

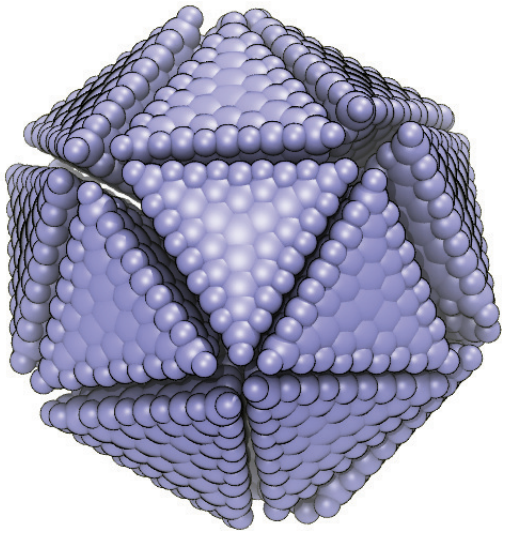


Figure 5.3: Each tetrahedron is composed of 81 balls. Sticky atches on the tetrahedra cause them to assemble into icosahedral clusters.

balls.

The filling problem can be expressed by the following two questions:

> Given a compact region G (having non-empty interior and no holes in the interior) and a fixed positive integer N, how can N balls be placed completely interior to G so as to maximize the total volume covered?

Overlaps of the N balls are permitted.

and

In general, for each fixed shape G, what is the best strategy for maximizing the fraction of volume covered by balls in G?

5.1 General Properties of the Medial Axis of G and Filling Solutions

5.1.1 Definitions and theorems

Let G be a compact (closed and bounded), simply-connected region with a non-empty interior. Let S be the boundary of G: $S = \delta\ G$. We use the notation $M(G)$ for the *medial axis* of G.

As in reference[126], we restrict S to have a tangent and curvature defined everywhere but at a finite number of points. At these points sided curvature exists from any direction along the boundary. For simplicity, we do not consider G with holes, nor G with a boundary that abuts itself.

The medial axis of an object, originally defined by Blum in reference[127], and also known as topological or medial skeleton, is the set of all points having more than one closest point on the object's boundary. The medial axis is a transformation of an n dimensional shape into an n-dimensional hypersurface defined by the locus of the centers of all n-balls that are tangent to the boundary at two or more points, where all such n-balls are contained in the shape. Such n-balls are *maximal balls*, where a maximal ball is also defined as a ball contained completely in G that is not a proper subset of any other ball also contained in G. A shape is the logical union of all its maximal n-balls.

The radius function of a medial axis is the radius of the maximal ball associated with each point. The medial axis and the radius function together are a complete

shape descriptor and can be used to reconstruct the shape.

Definition 5.1.1. Let R_N be a set containing N balls D_i with radii r_i and centers x_i that are completely contained in G. R_N is a *filling* solution of G. Let $\phi(R_N, G)$ be the fraction of G that is covered by R_N. ϕ is the filling value of the set R_N over G.

For a shape G, $\phi \leq 1$ by definition. The coverage ϕ is equal to unity for $N < \infty$ only if G is equivalent to a finite number of overlapping balls.

Definition 5.1.2. A set R_N is *all-covering* if, for all $D_i \in R_N$, $\phi(R_N - \{D_i\}, G) < \phi(R_N, G)$. In other words, each ball in R_N uniquely covers a non-zero volume of G.

Definition 5.1.3. A set R_N is an *optimal filling* solution of G if there is no other set R'_N that satisfies $\phi(R'_N, G) > \phi(R_N, G)$.

The function ϕ can be defined over the space of all R_N for a shape G and fixed positive N. Our objective is to find the set R_N with the maximum value of ϕ.

For an arbitrary shape G, there is no guarantee that such optimal solution sets must be unique. For example, Figure 1 shows how a long rectangle can have an infinite number of $N = 1$ solutions. All discs added to the middle of the rectangle have the same diameter. When sufficiently many discs have been added to the rectangle so that discs must overlap, this form of degeneracy disappears. Degenerate solutions also occur in symmetrical shapes with asymmetrical optimal filling solution sets, such as the $N = 2$ solution shown for the triangle in Figure 1 . This form of degeneracy does not disappear as $N \to \infty$. It is also possible for a shape with no symmetries to have two distinct optimal filling solution sets with the same filling area. The likelihood of two distinctly different sets of discs to both be optimal and cover the same measured area is unlikely, and thus this form of degeneracy is also likely to be extremely rare.

Theorem 5.1.1. *For G, there exists optimal filling solutions R_N that contain only maximal balls.*

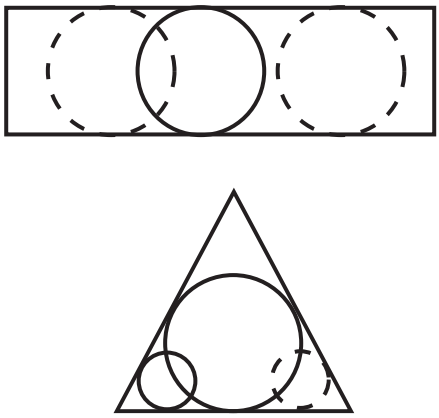


Figure 5.4: Two examples of degenerate solutions. In the case of the rectangle, a single maximal ball can be placed in infinitely many locations. For the symmetrical triangle, the asymmetrical solution can be reflected to generate a degenerate solution.

Proof. From any filling solution set that has a ball that is not on the medial axis, we can construct a solution set that contains only balls on the medial axis. Assume we have a solution set R_N that contains a ball D that is not tangent to S at any point. That ball is completely contained inside a concentric ball that is tangent to at least one point of S. And that ball is completely contained inside a larger cotangent ball that is also tangent to a second point of S. This last ball D' has its center on some part of the medial axis by construction and is thus a maximal ball. Let R'_N be the set of balls where D is replaced by D'. It must be that $\phi(R'_N, G) \geq \phi(R_N, G)$. So if R_N is an optimal filling solution, then so is R'_N. □

While Theorem 5.1.1 implies that filling solutions can be restricted to sets of maximal balls, it does not follow that optimal solutions must be composed of maximal balls for all shapes. Shapes that have boundaries with concave points of infinite curvature can have optimal filling solutions with non-maximal balls. Figure 5.5(a) shows such a shape. The outer boundary of the shape in Figure 5.5(a) is defined by four circular arcs. The dashed (green online) line is the medial axis of this shape. If the four discs at the extreme points of the medial axis have been placed, then there

is no need for the final disc placed inside the shape to be a maximal disc.

Even if restricted to optimal solutions of maximal balls, the entire medial axis need not be occupied as $N \to \infty$. Figure 5.5(b) is an example of a concave shape with two concave points. The portion of the medial axis between the two circle centers (red online) need not be occupied by disc centers to fill the shape as $N \to \infty$.

Theorem 5.1.2. *If S contains no concave points of infinite curvature, then optimal fillings R_N composed of maximal balls are also all-covering. Only fillings solution sets of maximal balls can be optimal. The entire medial axis is occupied for optimal filling solutions as $N \to \infty$.*

Proof. Assume there is an all-covering filling solution R_N for a shape G that contains a ball D that is not a maximal ball. The operations from Theorem 5.1.1 are used to construct a maximal ball D' on the medial axis from the ball D. The disc D' by construction, is tangent to the boundary of S in at least one location that D was not. Assuming that S has no concave points of infinite curvature (e.g a reflex vertex for a polytope), then the point of tangency is smooth and there is a small region around the point of tangency that D' covers that D did not (Figure 5.6). That region can only already have been covered by a ball in R_N if a ball contained in R_N of equal or larger radius, was tangent to S the same point. But since disc D', and therefore D, would be completely contained in that ball, then R_N would not be all-covering. So $\phi(R'_N, S)$ is strictly greater than $\phi(R_N, S)$. If each point on the boundary S is tangent to a maximal ball at a smooth point, then there is one maximal ball tangent to S at that point, so each point of S maps to a one and only one point on the medial axis. Also, a unique infinitesimal volume is covered by that maximal ball, so the entire medial axis must be occupied for the optimal fillings as $N \to \infty$. □

Theorem 5.1.1 shows that to construct filling solutions, the search space can be restricted to the space of maximal balls. Theorem 5.1.2 shows that for G with S

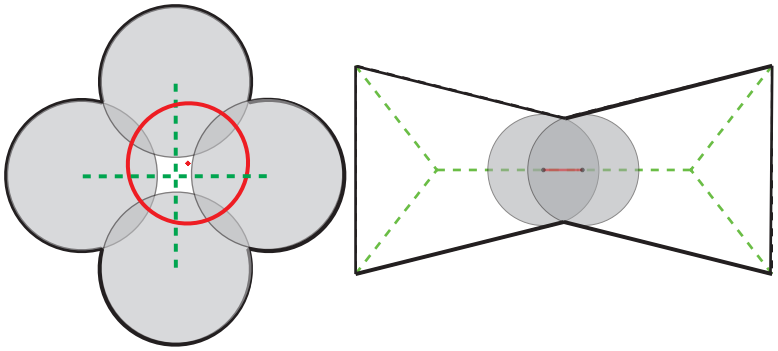


Figure 5.5: The construction on the left has optimal solutions without maximal balls. The center of the red disc need not be on the medial axis (dashed green) for the shape to be completely covered. The construction on the right need not have all of its medial axis (dashed green) filled. A disc added to the red portion fills no additional area.

Figure 5.6: If the point of tangency is smooth in any direction, the largest ball tangent to the point covers more volume than any smaller ball tangent to the point.

without concave points of infinite curvature, optimal filing solutions consist only of maximal balls. Finding optimal fillings has been reduced from find points in an $n+1$ dimensional space (disc center position and radius) to finding points on an n-dimensional hypersurface, or a problem of dimension $n-1$.

In practice, this hypersurface is better described as a set of bounded connected hypersurfaces. A planar shape, for example, has a medial axis that forms a planar graph, a connected set of curves that meet at points. In three dimensions, a medial axis is composed of sheets, seams, and junctions[128–130].

In the following sections we shall implicitly assume all R_N solution sets being discussed are all-covering.

5.1.2 Filling contribution of a single ball

The contribution of a single ball to the total filling is equal to the volume of the ball offset by its fractional share of the volume of any overlap with other balls. More explicitly, the contribution of a single ball is the volume it uniquely covers plus $1/i$ of the volume it shares with exactly i other balls. Let $V_i'(D_k)$ be the domain that a ball D_k shares with exactly i other balls, including itself. Let $\mathcal{M}_V(V)$ be the measure the volume of a domain. Note that $V_1'(D_k)$ is the domain uniquely covered by the ball D_k. The contribution $\mathcal{C}(D_k)$ is,

$$\mathcal{C}(D_k) = \sum_{i=1}^{\infty} \frac{1}{i} \mathcal{M}_V \left(V_i'(D_k) \right) \tag{5.1}$$

and,

$$\phi(R_N) = \frac{1}{\mathcal{M}_V(G)} \sum_{k=1}^{N} \mathcal{C}(D_k). \tag{5.2}$$

In general, if the overlap between ball P and Q is completely contained inside of the overlap between ball Q and R, then locally adding, removing, or locally displacing ball P cannot uncover any of that overlap volume. So if a ball P is added, removed, or locally displaced, the contribution of other balls may change but the only term that changes in the summed contributions of all the balls is $\mathcal{M}_V \left(V_1'(P) \right)$.

CHAPTER 6

Filling Solutions in 2D

6.1 Planar Shapes and Polygons

We now restrict the problem to regions, or shapes, G that are planar shapes, whose medial axis are the locus of the centers of maximal discs. Various algorithms exist to compute the medial axis of simple polygons and planar regions bounded by line segments, circular arcs, and general nonuniform rational B-splines[131–133].

We define the terminology and review the properties for a $M(G)$ of a planar shape introduced by Blum and Nagel[126]. $M(G)$ consists of connected subsets of points that form a 1-D planar graph. Most points of $M(G)$ are *normal points*, whose maximal disc is in contact with the boundary at two separate but contiguous sets of points. $M(G)$ also contains a finite number of *branch points*, each of which has a maximal disc in contact with the boundary at three or more separate but contiguous sets of points, and a finite number of *end points*, whose maximal disc is in contact with the boundary at only one contiguous set of points. For all but a finite numbers of discs, the contiguous set of points is a single point of contact. The contact point consists of more than just a single point if the radius of curvature the disc and boundary are the same. As long as G has no holes, then the graph $M(G)$ forms a tree with no loops. $M(G)$ can be divided into sets of contiguous normal points bounded by branch or end points, such that the division is unique, disjoint, and complete. Sets of contiguous normal points shall be referred to as a branch. The boundary of S can be

divided into parts associated with each branch by the intersection of S with the set of maximal discs defined over a branch. This division of S is also unique, disjoint, and complete[126]. The shape and radius function of any branch of $M(G)$ is determined by the *parents* of the branch, that is, the two contiguous sections of S associated with the given branch.

Given a contiguous set of normal points defining a branch, two separate but contiguous sections of S are associated with the branch. We note that given a sequence of maximal discs on the branch, their respective points of contact with S and their centers on the branch are traversed in the same order (assuming the branch is traversed in the correct direction relative to S).

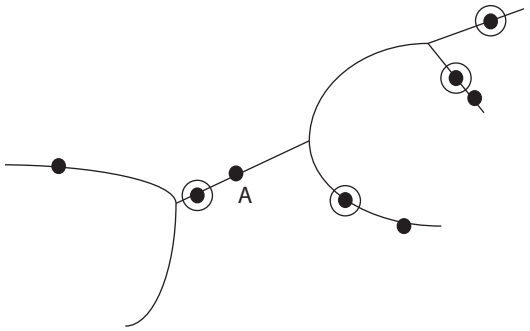


Figure 6.1: Each point represents the center of a maximal disc on $M(G)$. The neighbors of the disc A are circled.

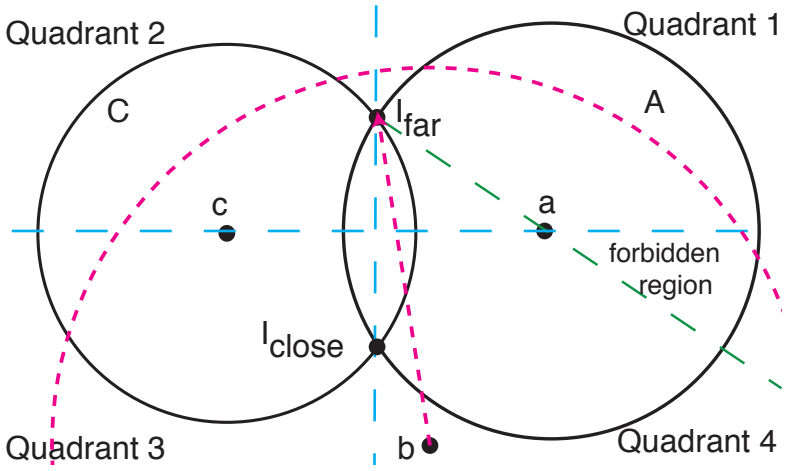
Definition 6.1.1. Given a filling solution R_N, and $D \in R_N$ for a $M(G)$ planar graph, we can define the *neighbors* of a maximal disc D, as the maximal discs whose centers are the closest along paths in $M(G)$ originating at the center of D.

In other words, if there is a path in $M(G)$ that connects the center of disc D to the center of disc D' without traversing another disc center, then disc D and disc D' are neighbors (Figure 6.1). For G with no holes, where $M(G)$ has no loops, there is only one path connecting any two disc centers. If a branch of the medial axis was populated with many maximal discs, most of the discs would have exactly two

neighbors, the disc to the left and right of them in sequence. If a branch point has connectivity n, then, in a densely populated $M(G)$ the disc closest to that branch point would have n neighbors. A disc can theoretically have as many neighbors as $M(G)$ has end points with connectivity 1.

Theorem 6.1.1. *Any overlap of D with any disc that is not a neighbor of disc D must be contained inside the overlap of disc D with one of its neighbors.*

Thus to measure $\mathcal{M}_V\left(A_1'(D)\right)$ for a disc D, only the position of the neighbors of D need be accounted for. To show the latter is true, we draw upon the properties of a planar medial axis.



Proof.

Figure 6.2: The construction of three discs A, B, and C that lie along a medial axis path.

Assume that the centers of three maximal discs A, B, and C, are on a path of $M(G)$. Let their centers be denoted by a, b, and c. The edges of discs A, B, and C are simply the circles with centers at a, b, and c of the same radii as the discs. Assuming $M(G)$ has no loops, then the path is the only path in $M(G)$ connecting the center of A to C. Suppose that there is an intersection between maximal disc A and C as constructed in Fig. 6.2. Two circles intersect in a region shaped like an asymmetric lens. Label the two points where the edge of disc A and C intersect

I_{far} and I_{close}. Divide the plane into four quadrants defined by the line connecting a and c and the line connecting I_{far} and I_{close}. Now construct disc B. Without loss of generality, we assume the center b is in the bottom right (Quadrant 4) of the figure. We also observe that b is a normal point or a branch point, but not an end point, as it is between a and c on a path. As disc B is constructed, there are restrictions on both where its center can be placed within Quadrant 4 and on its radius.

First, the point I_{far} must be contained in the disc B. We observe that as one is traversing the boundary S of the shape, the edge of A, B, and C must be encountered in the following order: a continuous set of disc A edge points, a continuous set of disc B edge points, a continuous set of disc C edge points, another continuous set of disc C edge points, another continuous set of disc B edge points, and another continuous set of disc A edge points. Other continuous sets of other disc edge points may interleave the sets specified, however, the specified order of encountering continuous sets of points of A, B, and C must still be followed. If the radius function along the medial axis path is redefined to exactly trace the boundary of A, B, and C, then this ordering must still hold. Therefore, the intersection points (labeled I_{far} and I_{close} in the figure below) between the edge of disc A and disc C must be contained in disc B.

Second, the radius of disc B cannot be larger than the distance between point b and the farthest point on the edge of disc A. Otherwise all of disc A will be inside disc B, making disc A not a maximal disc.

Third, the points on the edge of disc A not contained in disc B must include points not part of the edge of the asymmetric lens, or else disc A will be completely interior to disc B and disc C and not a maximal disc.

The remainder of the argument reduces to the following. Given a point b in Quadrant 4, what points on the edge of disc A are farther from point b than I_{far}? If a line is drawn connecting point b to point a then the line intersects the edge of disc A in two locations, one in Quadrant 1 or 2, the other in Quadrant 3 or 4.

The intersected edge point in Quadrant 1 or 2 is the farthest point on the edge of disc A to b. Tracing around the edge of disc A to the other intersection point, each point encountered is closer to point b than the last. Consider the region above a line intersecting I_{far} and point a in Quadrant 4. If point b was in this region, then only points on the intersection lens edge of disc A are farther from point b than I_{far} This violates the third rule above, so this region cannot contain b. If point b is restricted to the remaining region of Quadrant 4, then the distance to I_{far} is alway greater than any other point on the lens. Therefore, it is the case that the entire intersection overlap region of disc A and disc C is contained in disc B. □

The following theorem immediately derives from Theorem 6.1.1,

Theorem 6.1.2. *Let R_N be a the set of maximal discs on $M(G)$. Let $M(G)$ be divided into two locii of connected points P_1 and P_2 such that the only points P_1 and P_2 have in common are a finite set of points occupied by maximal discs $R_{N,boundary} \subset R_N$. Let $R_{N,1}$ be the maximal discs whose centers are are on P_1 but do not include $R_{N,boundary}$. Likewise, Let $R_{N,1}$ be the maximal discs whose centers are are on P_2 but do not include $R_{N,boundary}$. The area covered only by the set of discs in $R_{N,1}$ is the same for all possible $R_{N,2}$, and the area covered by only by the set of discs in $R_{N,1}$ is the same for all possible $R_{N,2}$.*

This theorem is illustrated in Figure 6.3.

6.1.1 Properties of an optimal planar filling

We make some observations about the function $\phi(R_N, G)$, where the discs of R_N have centers at points $x_i \in M(G)$.

Per equation 5.2, ϕ is a function of the area of discs and the area of overlap between discs. As the radius function is continuous over $M(G)$ and the area of overlap between shapes is continuous with respect to inflating or translating the shape, ϕ is

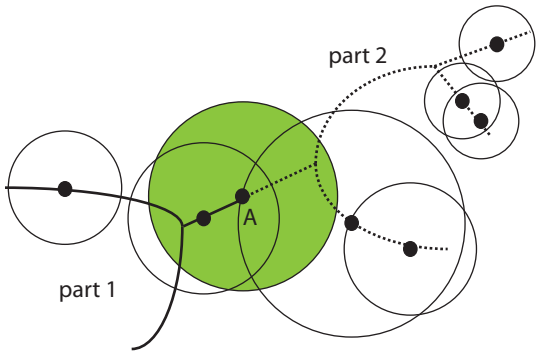


Figure 6.3: If disc A is kept fixed in position, then it divides $M(G)$ from Fig. 6.1 into two parts, indicated as solid and dashed lines. There is no intersection between discs on the two parts of $M(G)$ that is not covered completely by disc A per Theorem 6.1.1. The two parts, therefore, act as independent spaces.

a continuous function.

The change in ϕ due to moving a single disc center is equal to the change in the area uniquely covered by the disc. This uniquely covered area can be expressed as the area of the disc minus the overlap between the disc and its neighbors, $O_n(x, r)$, where x is the position of the center of the disc D and r is the radius of the disc. If the disc center is moved along a path parametrized by t, then

$$\frac{\partial \phi}{\partial t} = \frac{\partial \mathcal{M}_V\left(V_1'(D)\right)}{\partial t} = 2\pi r \frac{\partial r}{\partial t} - \left(\frac{\partial O_n(x,r)}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial O_n(x,r)}{\partial r}\frac{\partial r}{\partial t} \right). \tag{6.1}$$

The function $\frac{\partial \phi}{\partial t}$ is discontinuous when $\frac{\partial r}{\partial t}$, $\frac{\partial x}{\partial t}$, $\frac{\partial O_n(x,r)}{\partial x}$, or $\frac{\partial O_n(x,r)}{\partial r}$ is discontinuous. The radius function can be first-order discontinuous at a finite number of points, (e.g. branch points) as can $x(t)$ (e.g. branch points or any point of infinite curvature). At these points $\phi(t)$ can also be first-order discontinuous.

If a point of first-order discontinuity is also a local maximum (i.e. $\frac{\partial \phi}{\partial t}$ changes in sign at the point), then small displacements of the neighbors may shift the sided values of the discontinuity without affecting the sign change or shifting the position of the maxima. The point of first-order discontinuity creates a *center trap*. The local maximum at a center trap tends to stay stationary unless there are large rearrange-

ments of the points in its neighborhood. As a result, unlike other points on $M(G)$, a center trap tends to be a commonly occupied point in locally maximal filling solutions, optimal filling solutions, and even a fixed feature of filling solutions when N is large.

One type of point of first-order discontinuity is the point where one disc first contacts another. These are the points where $\frac{\partial O_n(x,r)}{\partial x}$ or $\frac{\partial O_n(x,r)}{\partial r}$ is discontinuous. Here the overlap area with another disc changes continuously from zero to positive. As a sign change cannot occur at this point, it cannot be an isolated local maximum and does not act as a trap.

We now introduce a new term, *junction point*.

Definition 6.1.2. A junction point is a point in $M(G)$ where, relative to some path in $M(G)$ that includes the point, either the radius function or the path itself is first-order discontinuous, or both. Junction points can act as center traps.

$\phi(t)$, therefore, is a continuous, piece-wise first-order continuous function. Insights into the structure of ϕ and $M(G)$ permit the design of an efficient heuristic with a high likelihood for finding an optimal filling solution.

6.1.2 Polygons

For a convex polygon, $M(G)$ is only composed only of line segments. For a simple polygon, $M(G)$ is composed of line segments and parabolic curves (Figure 6.4). For a convex polygon, the parents are always two straight edges. For a simple polygon, parents can be two straight edges, a straight edge and a reflex point, or two reflex points. The resultant branches of $M(G)$ and corresponding radius functions are then,

- *Case 1 (Figure 6.5a): two straight edges* A line segment with a linear (or constant) radius function. The path and radius function can be parameterized as $(x(t), y(t)) = \vec{A}t + \vec{B}$, and $r(t) = ct + r_0$, for $t \geq 0$, for constants $c, r_0 \geq 0$.

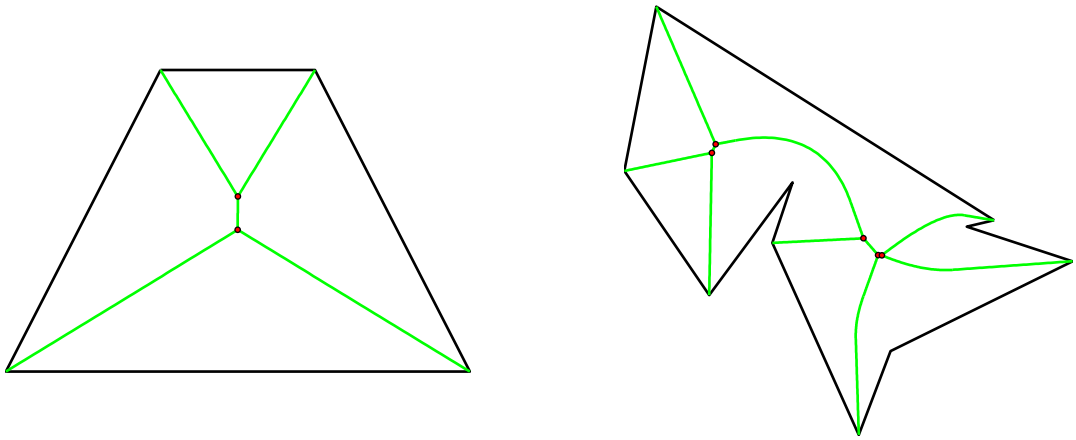


Figure 6.4: Shown are the M(G) of two simple polygons (green online). The dots (red online) represent junction points

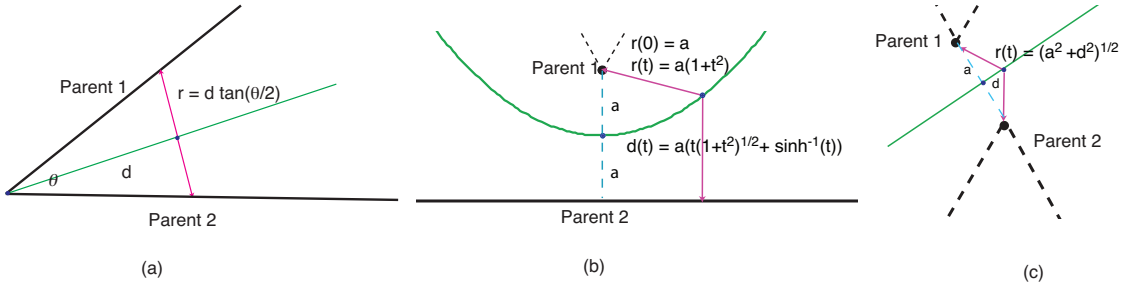


Figure 6.5: (a) Case 1, (b) Case 2, and (c) Case 3.

- *Case 2 (Figure 6.5b): a straight edge and a reflex point* A parabolic curve with a non-linear radius function. The path and the radius function can be parameterized as $(x(t), y(t)) = (2r_0t, r_0t^2)$ and $r(t) = r_0(t^2 + 1)$, where r_0 is the minimum of the radius function. The curvature can be parameterized as $\kappa(t) = (1/2r_0)\,(1 + t^2)^{-3/2}$.

- *Case 3 (Figure 6.5c): two reflex points* A line segment with a non-linear radius function. The path can again be parameterized as $(x(t), y(t)) = \vec{A}t + \vec{B}$. If $t = 0$ is the point on the medial axis halfway between the two reflex points, then $r(t) = \sqrt{a^2 + (a_xt^2 - b_x)^2 + (a_yt^2 - b_y)^2}$, where a is the distance from the halfway point to the reflex point, $\vec{A} = (a_x, a_y)$ and $\vec{B} = (b_x, b_y)$.

For simple polygons, the point where a line segment meets a parabolic curve (or a parabolic curve joins another parabolic curve) is not a branch point, but does involve a change in the geometry of $M(G)$[131]. In Figure 6.5, the three cases are illustrated with the parents labeled and the radius function shown as a function of the path or segment length, d, or of parameter t. Only branch points in simple polygons are junction points.

6.1.3 Center-occupied junction points and optimal solutions

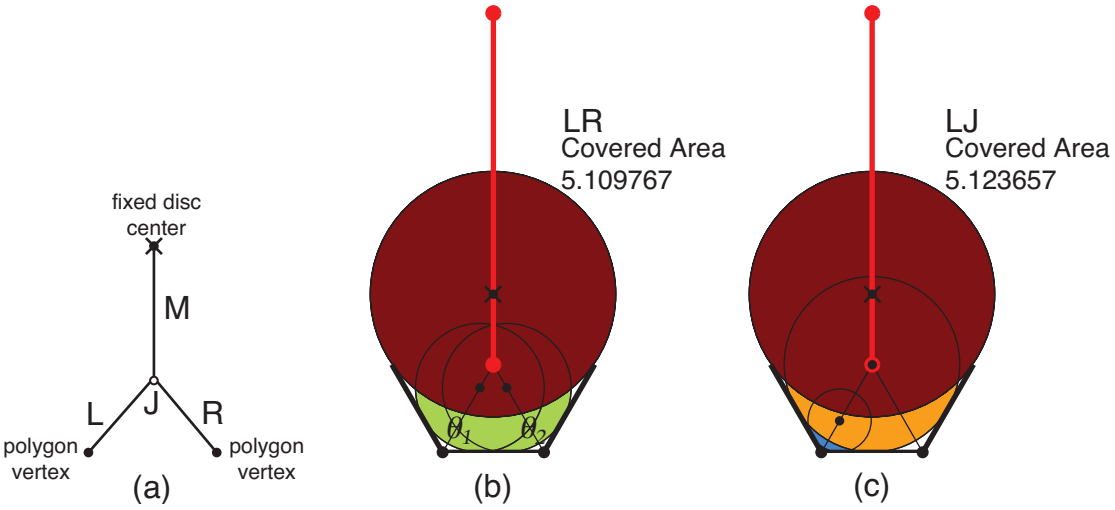


Figure 6.6: In (a) the labeled diagram of the isolated medial axis structure created by three connected polygon edges and a fixed disc is shown. For $\theta_1 = \theta_2 = 2\pi/3$ and $t = 0.2$, two locally maximal solutions, (b) a symmetrical solution with a disc on the L and R branch and (c) an occupied junction LJ solution are shown. The second solution is the global maximum.

The strategy of the optimization algorithms in the subsequent sections is to find the optimal filling solution, or the global maximum of ϕ, by generating a population of local maxima. Junction points are special points that need to be handled separate from branches by a maxima finding search method.

To demonstrate the diverse landscape of these maxima, and the role of junction points therein, we employ the following numerical experiment.

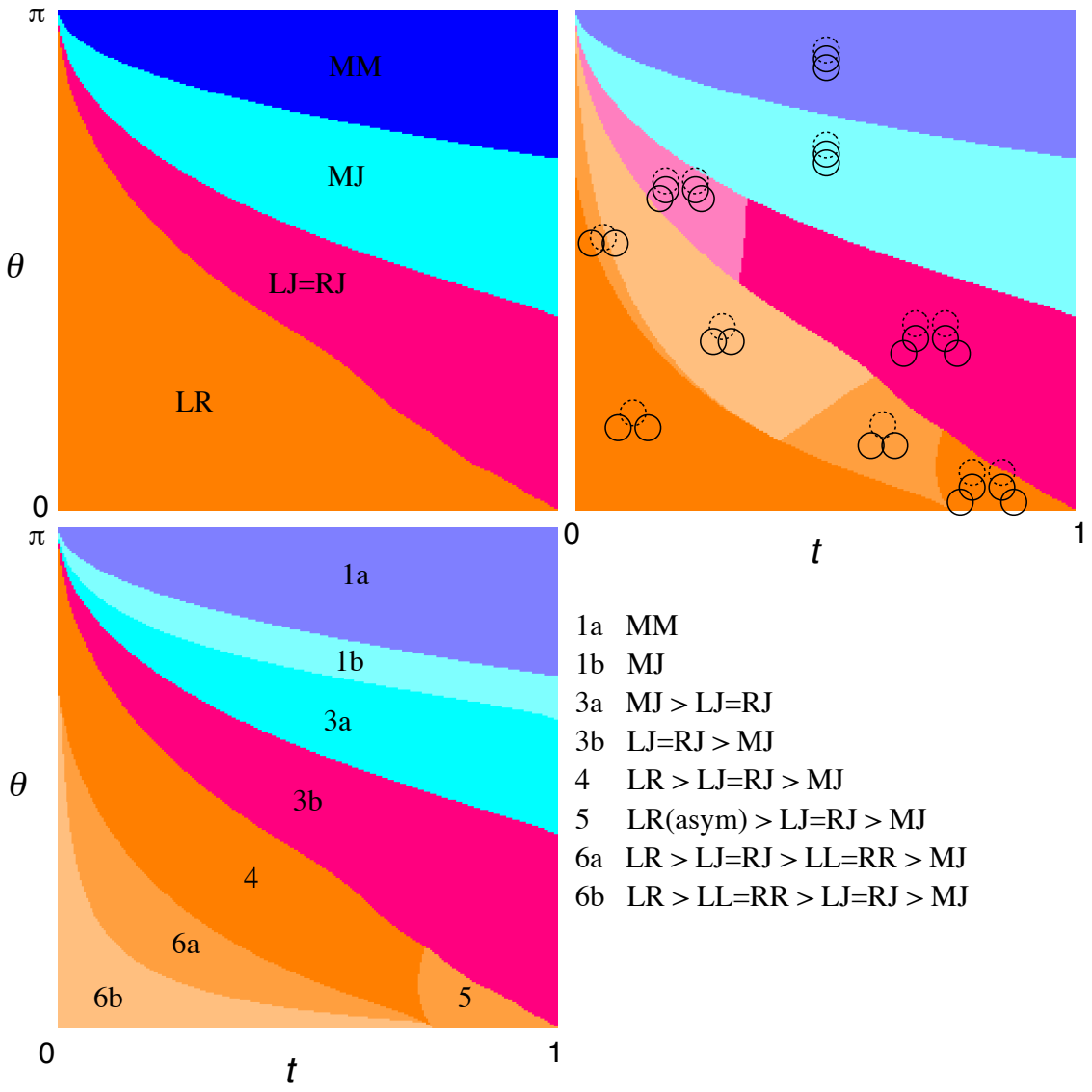


Figure 6.7: Case (A): For the constructed problem of Figure 6.6, $\theta = \theta_1 = \theta_2$, on the top left is shown the type of the global maximum as a function of θ and t. On the right the topological structure of the global maximum is shown. On the bottom left, the number and form of the maxima in each part of the phase diagram is indicated.

A section of a polygon G is constructed from a ray, a segment, and a ray. The medial axis of this shape is two straight branches (parents are a ray and the segment) and a straight branch that is a ray (parents are the two rays) that meet at a junction point, shown in Figure 6.6. The ray branch is parameterized by setting $t = 1$ at the point on the ray branch where a maximal disc would be just tangent to a disc sitting

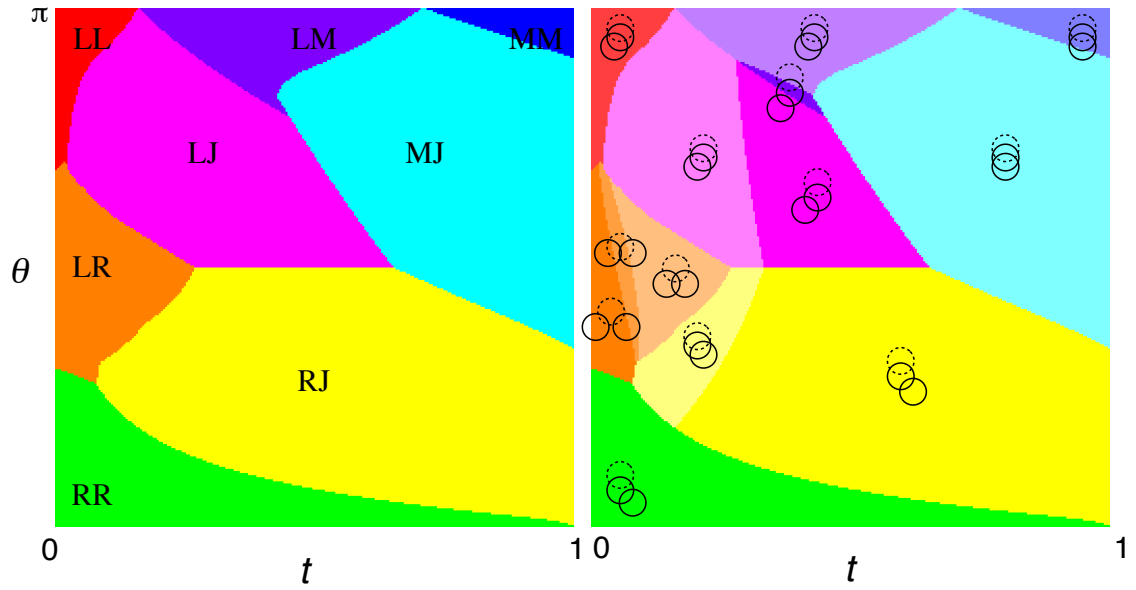


Figure 6.8: Case (B): For the constructed problem of Figure 6.6, but with $\theta_1 = \pi/2$, $\theta = \theta_2$, on the left is shown the type of the global maximum as a function of θ and t. On the right the topological structure of the global maximum is shown.

on the trap junction point, and $t = 0$ be the trap junction point itself. For $0 \leq t \leq 1$ and θ, we fix a disc on the t position on the ray branch. This region of the polygon is now completely isolated from the rest of polygon per Theorem 6.1.2. We now search for all the local maxima of adding two discs to the region by using simple gradient methods. We perform this search for two cases, (A) angles θ_1 and θ_2 between each ray and the segment are set identically to θ, and (B) $\theta_1 = \pi/2$ and $\theta_2 = \theta$. In Figure 6.6b and 6.6c, an example case (A) is shown for $\theta_1 = \theta_2 = 2\pi/3$ and $t = 0.2$. The red circle with an $\times$ at its center is the fixed center. Figure 6.6a shows the graph structure defined by the medial axis of this construction, with branches labeled L, R, and M, connected by junction point J. All local maxima of this construct can be classified by the branches or junction point that contain a disc.

Figure 6.6b and 6.6c are two local maxima. The figure 6.6b is a local maximum without an occupied junction point or an LR maxima. Figure 6.6c is a local maxima

with an occupied junction, or an LJ maxima. Because for case (A), $\theta_1 = \theta_2$, the maxima of this case often are symmetrical or degenerate. Consider, for example, an LR maxima. If the solution is symmetrical (i.e. the left and right disc have the same radius) then it represents a single maximum of this type. However, if the LR maximum is not symmetrical (i.e. left and right disc have different radii), then there are two degenerate maxima of this type. For case (A), the LJ type maximum is always symmetrically equivalent to an RJ maximum. In top left of Figure 6.7 a diagram of the form and region of the global maximum are shown for case (A) for $\theta = 0.0$ to π radians and $t = 0$ to 1. The junction point is occupied for a large region of the phase diagram (LJ=RJ and MJ). In the top right of Figure 6.7 the topological form of the global maxima is shown, each shaded a different color. The dotted circle is the fixed disc. The MM and MJ each have one form of topological intersection between the three discs, the LJ=RJ has two forms, and the LR region has five. The global maximum found in the LR region is symmetrical everywhere except for high t. In this region, one of the L or R discs is larger and overlaps the fixed disc, while the other is smaller and does not. In the bottom left of Figure 6.7 the number and type of local maxima found in each part of the diagram is shown. The number of maxima range from one to six. On the bottom right is a key for indicating which types of local maxima can be found in each region. For example, in region 6b, there are six local maxima. The symmetrical LR maximum has a higher filling than a LL maximum, which is identical to an RR maximum. The LL maximum has a higher filling than an LJ maximum which is identical to an RJ maximum. An MJ maximum has the lowest filling of all the local maxima in this region. Notably, the MJ local maximum is present everywhere on the phase diagram, except in region 1a, where MM is the single only maximum.

In the left of Figure 6.8 the form of the global maximum is shown for case (B) where $\theta_1 = \pi/2$ and $\theta_2 = \theta = 0.0$ to π radians and $t = 0.0$ to 1. For case (B), there

is no symmetrical solution region (aside from a region of zero area where $\theta_2 = \pi/2$ The space is divided into eight types of global maximum. The only global maximum solution form that is not found is the RM solution type. On the right of Figure 6.8, the topological form of the global maximum is shown. The solution regions MM, MJ, RR, and LL have a single topological type, the solution regions LJ, RM, and RJ each have two, and the solution region LJ has three. Interestingly, the case (B) LR region has two fewer topological types that the case (A) LR region. For case (B), the mapping of the number of local maxima is not shown, as the landscape proved extremely complicated and difficult to map.

The number of maxima and solution type landscapes in both Figure 6.7 and 6.8 are surprisingly complicated. While the boundaries between different solution type regions may correspond to some form of analytical expression, the expression is not known. If the number of possible types is small, it is simplest to check each possible solution type to find the global maximum. The junction point on $M(G)$ is occupied in the global maximum solution a large fraction of the time. This numerical experiment justifies the treatment of junction points as special points in the medial axis point set.

6.2 Algorithms for Generating Filling Solutions

We know of no analytical method for finding the optimal filling solution for a given N. Instead we introduce two algorithms that explore the objective function landscape of ϕ to search for the global maximum.

6.2.1 Genetic algorithm

A genetic algorithm (GA) is employed to find the optimal filling solution for polygons. The benefit of the GA is that it uses a minimal number mathematical assumptions about the space of the filling solutions, although the computational time

required is prohibitively long for $N > 20$.

6.2.1.1 Algorithm Description

GAs start with an initial random population of solutions which are combined and mutated until no better solutions are found after a fixed number of iterations.

For this implementation, $100N$ to $400N$ population members are initialized. Each member is an ordered set of coordinates of N discs. If a disc is randomly generated outside the polygon, it is moved inside.

Without loss of generality, but with dramatic improvement of the efficiency and accuracy, the GA assumes that solutions will consist of maximal discs and attempts to construct them. First, the radius of each disc is grown to touch the nearest edge in the polygon. Second, if it can be determined that the disc is in a corner of the polygon, e.g. the nearby medial axis is a straight branch terminating in an end point, then the disc is moved to the nearest point on the medial axis, constructed by generating the bisector of the corner's internal angle.

These constraints are applied to the entire population and then ϕ is computed for each member. The population is then sorted by ϕ to produce a list of ranked solutions from best to worst. Members of generation g are randomly chosen as parents for generation $g + 1$. The relative probability p of a given member being chosen is weighted by its rank r, $p = 1/\sqrt{r}$. The next generation of members is created from the current generation as follows. (1) *Best* The best members, unmodified, are included. (2) *Mutation* One "parent" member is randomly chosen and is randomly mutated by either moving a disc completely randomly, displacing a disc up to 1/2 the polygon's width, w, displacing a disc up to 1/200th of w, or moving a disc to a junction point. The mutated "child" member is included. (3) *Crossover* Two parent members are selected and their discs are spatially sorted by $x * w + y$, where (x, y) is the position of a disc. A crossover point $C \in [1, N]$ is randomly determined. The child member

contains the discs with indices from 1 to C from the first parent and the discs from $C+1$ to N from the second. The child member is then included.

The fraction of the next generation generated by method (1), (2), and (3) can be modified to improve the outcome of the algorithm.

Even with the GA's capability of exploring many local maxima to find the global, it can still get trapped in a local maximum. The GA is run with different random number seeds ten or more times for each shape and value of N. The best of the best solutions obtained over all these runs is selected as the GA's final answer.

The mathematical assumptions the GA uses are first, per Theorem 5.1.1, that solutions should consist of maximal discs. Second, the GA also randomly places discs centers on junction points. As discussed in sections 6.1.1 and 6.1.3, junction points can act as center traps, are occupied as part of many local maxima. However, the basin of attraction around the junction point can be small enough that without including a random mutation to try occupy the junction, a solution with an occupied junction point can be too rarely found by the GA.

6.2.2 Heuristic algorithm for filling a polygon

In this section we introduce a heuristic algorithm that generates a putatively optimal N filling solution, by exploiting the properties of the $M(G)$ structure to generate a collection of unique local maxima. If enough are generated, the global maximum is among them. For the two-dimensional filling problem, we propose a local maxima generating strategy whereby centers are distributed onto $M(G)$ and the local filling maxima for that initial guess is found by simple gradient methods (e.g. active set or sequential quadratic programming optimization schemes). We also propose a method for reducing the number of such distributions needed to find an optimal solution by using the $N-1$ filling solution to generate the N filling solution.

The first step is to intelligently divide up $M(G)$. The medial axis is divided

into K pieces, maximally long branch sections with monotonically increasing radius functions and the junction points connecting them. To generate these pieces for a polygon, Case (2) and Case (3) may need to be divided into separate sections and joined with other branch sections. The medial axes of the left and right polygons depicted in Figure 6.4 are composed of seven and seventeen pieces respectively.

Definition 6.2.1. A *way*, W, is a distribution of N discs over the K pieces (branch sections and junction points), $W = \{n_i\}_1^K$ where $N = \sum_i^K n_i$ and $n_i \in \mathbb{N}$. If the i-th piece is a junction point then $n_i \in \{0, 1\}$.

Conjecture 6.2.1. There is at most one local maxima per *way*.

If Conjecture 6.2.1 holds, then to find the optimal N filling solution, a maximum must be generated for every way of N discs and K pieces. If J is the number of pieces that are junction points, then the number of maxima to be searched is of order $O\left(N^{K-J-1}\right)$ (see A-4).

Conjecture 6.2.2. Given the optimal *way* of distributing $N-1$ discs, $\{n'_i\}_1^K$, the optimal *way* of distributing N discs is nearby, where nearby means, $\sum_1^K \mid n_i - n'_i \mid$ is small, and that if the discs assigned to a given piece is decreased, the pieces that have discs increased have a minimal distance (counted by number of connecting pieces) to the decreased piece.

Conjecture 6.2.3. For a given G and $M(G)$, there is an N', such that for $N \geq N'$, the junction points are always occupied in the optimal filling solutions.

Given the *way* of the $N-1$ filling solution, the heuristic generates the local maxima of the nearby *ways* using a local maxima finding technique. The best local maximum found is presumed to be the optimal N filling solution for the shape. This heuristic is made more efficient by taking advantage of center occupied junction points and the dependence of the filling function on the nearest neighbors. We implement this

heuristic for polygons, which have a limited set of medial axis parameterized pieces to be handled.

6.2.2.1 Detailed Description of Heuristic

Following is a more detailed description of the Heuristic Algorithm.

Auxillary Algorithms The following sub-algorithms are needed to deploy the HA.

1. *Generating and Dividing $M(G)$ into K pieces.* The medial axis of the polygon is generated*. Parabolic curves (Case 2) and straight curves (Case 3) that include a minimum in the radius function are split at the minimum. The split branches are then recombined to form maximally long paths with monotonically increasing radius functions. Branch points are separated from branch pieces as junction point pieces.

2. *Calculating the Area of a Union of Discs.* The total area of the union of the discs is determined analytically by dividing the space into intersection regions defined by boundary arcs and calculating the area of each region[135]. The method can be applied to the entire set of discs, or far more efficiently, by dividing the calculation over the discs on each piece. In this latter method, first the area of the union of discs on each piece is calculated and summed. This sum over-counts the overlaps between union of discs of different pieces. Second, the overlap between the disc at the end of a piece and its the neighboring discs on other pieces, is subtracted from the total, once for each time it was over-counted. This latter method is more complicated, but also more computationally efficient because the areas of smaller sets of discs are calculated.

*using the matlab software package MatlabMedialAxis-Version 2.0 provided by Suresh Krishnan[134]

3. *Partitioning a Graph by Occupied Junction Points.* By taking advantage of regions of the $M(G)$ graph isolated by occupied junction points, per Theorem 6.1.2, filling solutions can be divided into solutions of independent sub-spaces of $M(G)$. Using the topology of the $M(G)$ graph and a set of maximal discs R_N, this algorithm step divides the graph into *parts*, or sets of pieces isolated from each other by occupied junction points.

4. *Finding the Local Maxima.* A solution set of discs can be uniquely represented by the way $W = \{n_i\}_1^K$ and a set of parameters $\{t_{i,j}\}$ where $i \in [1, n_i]$ and $j \in [1, K]$ and $t_{i,j} \in [0, 1]$† Given an initial guess way and a parameter set, an optimization method (e.g. active set or sequential quadratic programming optimization schemes[136]) is applied by using ϕ as the objective function. If the initial guess includes a trial disc insertion into a piece of a given part of $M(G)$, then all $t_{i,j}$ parameters of the part are free parameters. All $t_{i,j}$ parameters outside the part are fixed.

Heuristic Algorithm Given the $N-1$ solution:

I. For each part of the $M(G)$ graph isolated by occupied junction points, a new disc is inserted into each piece and the best solution for the part is found.

II. For each occupied junction point of the $N-1$ solution, the disc center is removed from the junction point and a initial guesses are generated by inserting two discs into nearby pieces and finding the local maximum. The more combinations of nearby pieces are tried, the larger a neighborhood is considered.

III. The N Solution is constructed from the best trial solution found. If a piece k has a parameter value $t = 0$ or 1, indicating that the junction, piece k', at the end of the piece has been occupied, then the disc is moved to junction point piece. If a

†if a piece terminates in a junction point at t=0 or 1 or both, then $t_{i,j} \in (0, 1]$, $t_{i,j} \in [0, 1)$, or $t_{i,j} \in (0, 1)$, respectively.

solution was generated for a part of $M(G)$ and not included in the best solution, and the part is found in both the N solution and the $N-1$ solution, then the solution is cached.

6.2.2.2 Algorithmic efficiency

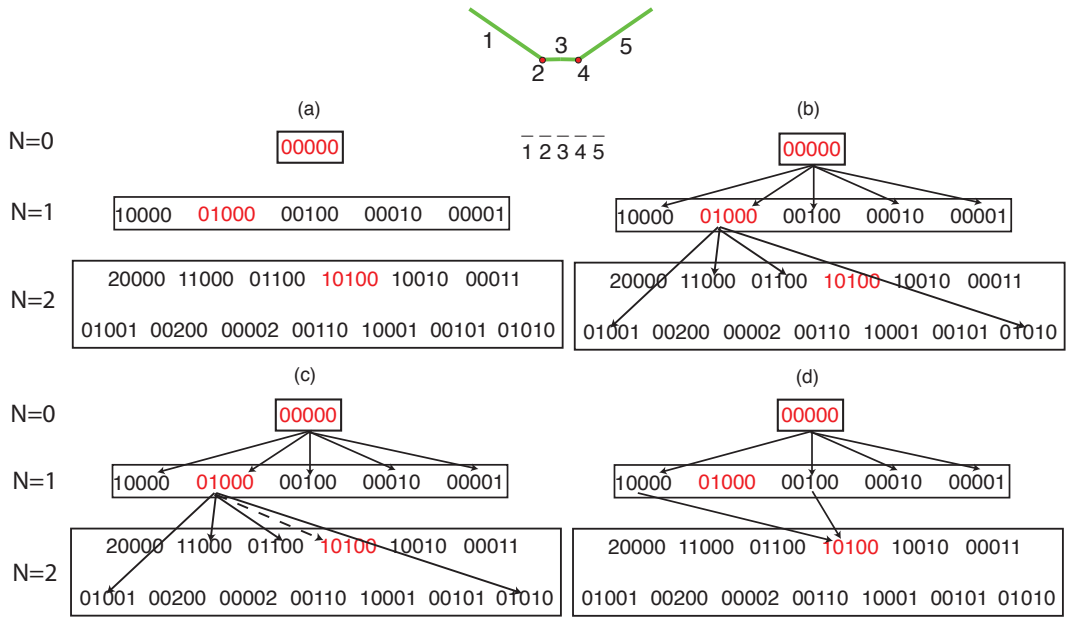


Figure 6.9: At the top of the figure is a medial axis with 5 pieces, three branches and two junction points. (a) The full table of *ways* is shown for N=0, 1, and 2. In (b) the search space is reduced using the greedy assumption that the next best solution is related to the last best solution. (c) We also add searches that deoccupy junction points and inserts discs onto nearby branches. (d) If the best 1-way was not searched, two of the four remaining 1-ways would have searched the best 2-way on the next iteration.

Using Greediness to Reduce the Search Space The heuristic exploits Conjecture 6.2.2 to reduce the number of ways to search for local maxima (i.e. number of initial guesses) from $O\left(N^{K-J-1}\right)$ to $O(N(K+J))$. Figure 6.9 depicts a hypothetical medial axis with two junction points and three branch pieces. Figure 6.9(a) shows a table of all possible ways for N=0, 1, and 2. Rather than search each way of $N=2$,

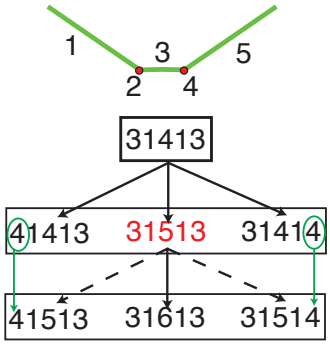


Figure 6.10: Assume that the junction points 2 and 4 stay occupied. To generate the N=13 solution, three ways are searched for a local maximum. To generate N=14, only one additional way, 31613, needs to be searched. The ways 41513 and 31514, can be created by combining the search of branch 1 and 5 with the solution of branch 3 for N=13. The occupied junction points isolate the solutions on each branch from solutions on the rest of the medial axis.

a reduced set is searched. That set is generated as follows, given the best $N-1$ way, one disc is added to each piece (that is not an occupied junction) Figure 6.9(b). Then, for each occupied junction point in the $N-1$ best way, the junction point is deoccupied and disc is inserted into two of the branches nearby the junction, Figure 6.9(c). The maximum number of ways that will be searched, given the best found $N-1$ way, is $K+AJ$, where A is a constant dependent on how large a neighborhood of a junction point one chooses. While this heuristic is not guaranteed to find optimal solutions, it finds a putatively optimal N filing solutions with only a $O(N(K+J))$ number of searches.

For example, for a triangle with K=4, finding the best arrangement of $N = 10$ discs means searching 121 ways, and the best arrangement of $N = 100$ discs requires searching 10,201 ways. By using a heuristic that exploits Conjecture 6.2.2, to finding the best arrangement of $N = 10$ discs requires searching only 70 ways, and for $N = 100$ discs, only 700 ways.

Applying optimization techniques to N' discs where $N' < N$ The computational effort to calculate the analytically exact area of the union of a set of N discs is super-linear in N, as can be optimization methods of N parameters. The exact order of the computational effort is dependent on the arrangements of the discs and the details and convergence rate of the optimization algorithm. The greedy heuristic of section 6.2.2.2 not only requires searching fewer ways, but also mostly searches ways of N' discs where $N' < N$. This significantly improves the computational efficiency of finding a solution.

Efficiently Sub-Dividing the Search Space The heuristic also improves efficiency by exploiting the properties of the solution space per Theorem 6.1.1 and 6.1.2 and the behavior of center traps as discussed in section 6.1.1.

When a junction point is occupied by a disc center in the $N-1$ solution, the center is usually trapped and the phase space of centers can be divided into independent sub-spaces. If it is known (or guessed) that the best solution for N also includes a center at the junction, then the sub-parts of $M(G)$ connected only by the junction point can be searched independently. Per Theorem 6.1.2, rearrangements of centers in one sub-part cannot affect the best arrangement of centers in another if they are connected only by a center-occupied junction.

This means that searches can be performed on a subset of the N discs (efficient per section 6.2.2.2) and that solutions of independent sub-spaces of $M(G)$ can be cached. Figure 6.10 shows how, when junction points are presumed to remain occupied, only one additional search of a way is needed to generate the next putatively optimal N filling solution. Per Conjecture 6.2.3, at sufficiently large N, junction points are occupied. This implies that for large N, generating the optimal N filling solution from the optimal $N-1$ filling solution requires a search of only one additional way, reducing the complexity of the HA to $O(N)$ searches.

6.2.2.3 Self-correcting

One weakness of a method that uses the $N-1$ solution to find the N solution is that if the optimal N' solution is not found, all solutions for $N > N'$ may not be optimal as well. However, in most cases where the HA does not find the optimal N' filling solution, by some $N > N'$, the HA will be generating the optimal solution again. That is, even if the wrong solution is found, the solution finding method will tend to self-correct at a higher N.

In Figure 6.9(d), for example, if the optimal N=1 way was omitted from the search, the optimal N=2 way would still be searched by two of four alternate N=1 ways. If, because of the neighborhood size chosen for deoccupying junction points, the heuristic fails to search the optimal way, per Conjecture 6.2.3 the junction point will eventually be occupied again in the optimal solution. Thus the optimal solution and the heuristic's solution will likely match again.

6.2.2.4 When the heuristic algorithm fails

Even if the Conjectures 6.2.1, 6.2.2, and 6.2.3 above hold, in practice this Heuristic Algorithm may still fail to find the optimal solution for the following reasons.

(1) Assuming a *way* has no local maximum While searching for the local maximum associated with a *way*, it is common to generate the local maximum of a nearby *way* instead (e.g a junction point becomes occupied). This leads to the conclusion that the *way* has no local maximum. However, the search may simply have been initiated outside the basin of attraction of the local maximum of the *way*.

(2) Searching in too small a neighborhood As discussed above, some optimal N solutions require looking in a larger neighborhood of the $N-1$ solution. Tradeoffs that balance confidence in finding the optimal solution against the computational

cost of searching larger neighborhoods may result in optimal solutions being missed.

(3) Solutions are only as good as the optimization method applied Lastly, local maxima finding techniques can have trouble converging. This is not a failure of the Heuristic Algorithm, per se, but occasionally affects the HA solution. Switching which nonlinear constrained minimization optimization technique is being applied generally solves the problem.

6.2.3 Heuristic vs. genetic algorithm filling solutions

To assess the capability of the heuristic algorithm (HA) *vs.* the GA, solutions were generated for N=1 to 21 for a selection of five convex polygons and twenty-one concave polygons. The putative best solutions produced by this HA match the solutions generated by the GA well. Specifically, the HA almost always produces solutions of the same *way* as the GA. The gradient optimization technique employed by the HA is usually better at converging to a final set of disc positions for a given *way* than the GA. On rare occasions the HA and GA find different *way*. When the HA *way* is better, the GA has usually become trapped in the wrong local maximum. When the GA *way* is a better solution, we find that the *way* was outside the neighborhood that was searched by the HA. Examples of filling solutions are shown in Figure 6.11.

	HA and GA Way Match	Best Way: HA	Best Way: GA	Best ϕ : HA
Convex	98.1%	1.9%	0%	100%
Concave	92.97%	3.4%	3.63%	96.37%

6.3 Continuum Solutions in a Polygon

An alternate perspective for understanding filling solutions is to ask how to optimally distribute infinitely many centers over $M(G)$ and how optimal solutions converge to the total area. As discussed in reference[83], the continuum solution can be

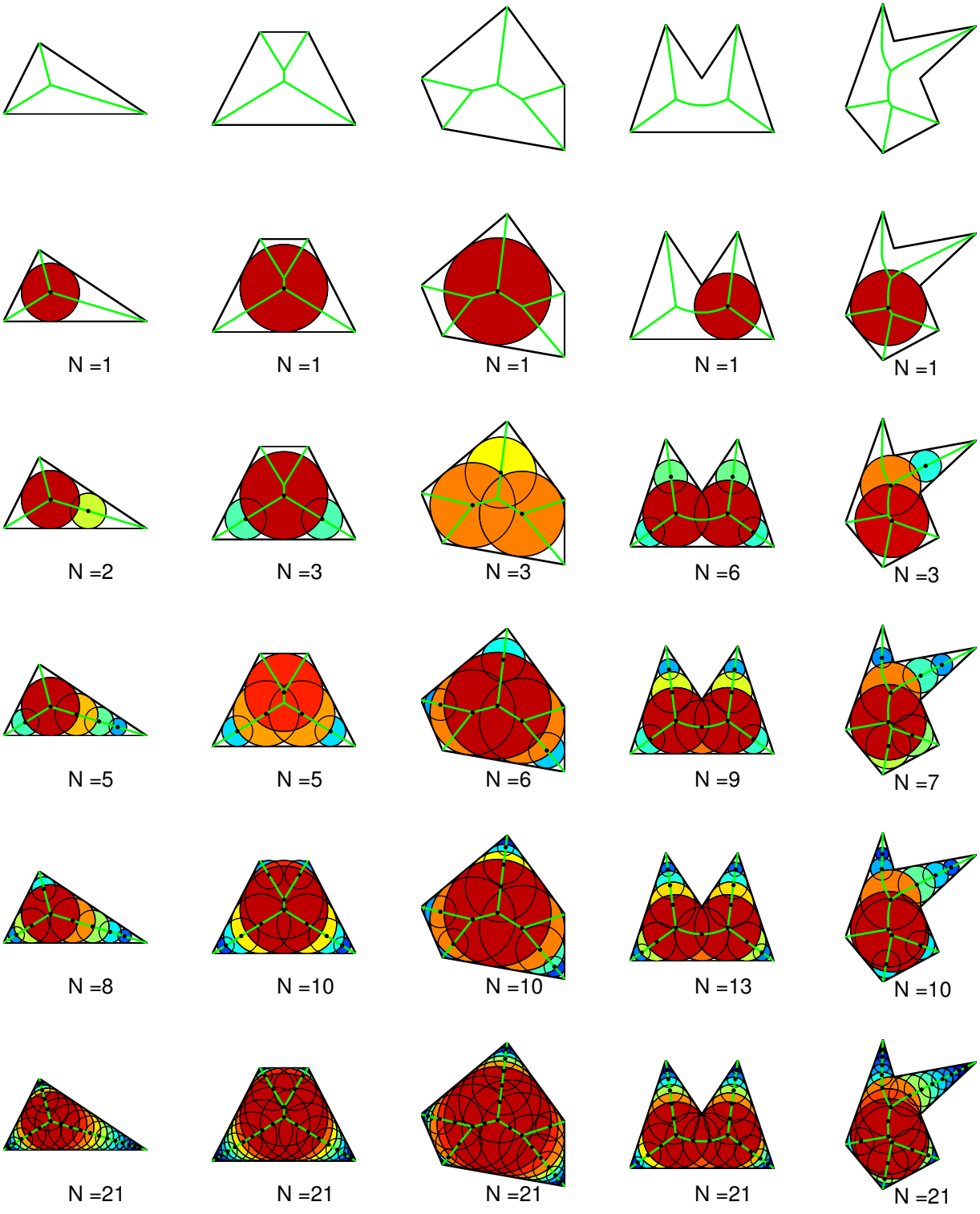


Figure 6.11: Examples of the optimal filling solutions of three convex and two concave polygons for N=1-21. The top row shows the medial axis of each polygon.

solved exactly for simple polygons by analyzing the three types of branches found in simple polygons (Case (1), (2) and (3) of Figure 6.5). For the Case (3) type, no disc centered on such a curve fills any more area than what is filled by placing two discs at the ends of the curve. So in optimal solutions, Case (3) type curves are empty except for their ends.

Let $\rho(t)$ represent the density of centers along a parameterized path of $M(G)$, $r(t)$ be the radius function, and $\kappa(t)$ be the local curvature of the path, where $(x(t), y(t))$ is the parameterization $t \in [t_a, t_b]$. Given an expression for the unfilled area A_i along the path i of $M(G)$ of the form,

$$A_i = \int_{t_a}^{t_b} C_i(\kappa, r', r) \frac{dt}{\rho^2}, \tag{6.2}$$

where C_i is a function to be determined, we would like to determine the function ρ that minimizes this area constrained by

$$N = \int_{t_a}^{t_b} \rho dt. \tag{6.3}$$

Note that if we sum the unfilled areas A_i over all of $M(G)$, then filling value $\phi = 1 - \sum(A_i/A_G)$, where A_G is the area of G. This variational problem can be solved by forming the Lagrangian

$$\mathcal{L}[\rho(t); \lambda] = \int_{t_a}^{t_b} \left(C_i(\kappa, r', r) \frac{1}{\rho^2} + \lambda \rho \right) dt \tag{6.4}$$

and taking the pointwise derivative with respect to $\rho(t)$,

$$\frac{\partial \mathcal{L}}{\partial \rho} = \int_{t_a}^{t_b} \left(\frac{-2C_i(\kappa, r', r)}{\rho^3} + \frac{\partial}{\rho^2 \partial \rho} C_i(\kappa, r', r) + \lambda \right) \delta(t - \tau) dt. \tag{6.5}$$

This relationship is satisfied by functions ρ that satisfy

$$-2C_i(\kappa, r', r) + \rho \frac{\partial}{\partial \rho} C_i(\kappa, r', r) + \rho^3 \lambda = 0 \tag{6.6}$$

Solutions of the form

$$\rho = \left(\frac{C_i(\kappa, r', r)}{\lambda} \right)^{1/3} \tag{6.7}$$

satisfy this equation.

For Case (1), where $(x(t), y(t)) = At + B$, $r = r_0 t$, and $t_a > 0$,

$$C = \left(1 - r'^2\right)^{3/2} / (12r) \tag{6.8}$$

as shown in Appendix A-1. It follows that $\rho = \propto r^{-1/3}$.

For Case (2), where $(x(t), y(t)) = (2r_0 t, r_0 t^2)$, $r = r_0(t^2 + 1)$, r_0 is the minimum of the radius function, and $\kappa(t) = (2r_0)^{-1} (1 + t^2)^{-3/2}$,

$$C = \frac{1}{12} \left(\frac{r_0 \kappa}{r} \right) = \frac{1}{24 r_0} \left(\frac{1}{1 + t^2} \right)^{5/2} \tag{6.9}$$

as shown in Appendix A-2. It follows that $\rho = \rho_0 \left(\frac{1}{1+t^2} \right)^{5/6}$ or $\rho \propto r^{-5/6}$.

For both Case (1) and Case (2), the distribution of centers follows a power law with respect to the local radius function. Centers on $M(G)$ will be distributed more densely where the radius function is smaller. Given $\rho = \rho_0 r^{-\alpha}$, for $\alpha = 1/3$ or $5/6$, ρ_0 can be determined from Equation 6.3,

$$\rho_0 = N \left(\int_{t_b}^{t_a} r^{-\alpha} dt \right)^{-1} \tag{6.10}$$

$$\rho_0 = N / R_0. \tag{6.11}$$

R_0 is then a constant determined by the radius function of branch section of $M(G)$.

For Case (1), the distribution of centers on the medial axis path is also *scale-free*.

The distribution of centers also follows a power law with respect to the distance from the vertex (where $t = 0$) of the polygon.

Equation 6.2 becomes,

$$A = \frac{1}{N^2} \int_{t_a}^{t_b} R_0^2 C(\kappa, r', r) dt = \frac{1}{N^2} \mathcal{C}. \tag{6.12}$$

Thus, in the continuum limit the filling converges to the area of the shape with an asymptotic error proportional to N^{-2} for ideally distributed centers. We presume that all shapes that can be approximated by simple polygons with an increasing number of sides also converge with an N^{-2} error term.

If we divide $M(G)$ into k branch sections we can predict what fraction of the discs (N_i/N) will be distributed over each branch i as $N \to \infty$.

$$A = \sum_1^k A_i(N_i) \tag{6.13}$$

$$N = \sum_1^k N_i \tag{6.14}$$

Since we have distributed our discs optimally, we can treat $A_i(N)$ as continuous function and

$$\frac{\partial A_i}{\partial N_i} - \frac{\partial A_j}{\partial N_j} = 0, \forall j \neq i \tag{6.15}$$

Arbitrarily setting $j = k$,

$$\frac{\partial A_i}{\partial N_i} - \frac{\partial A_k}{\partial N_k} = -2\frac{\mathcal{C}_i}{N_i^3} + 2\frac{\mathcal{C}_k}{N_k^3} = 0 \tag{6.16}$$

and

$$N_i = \left(\frac{\mathcal{C}_i}{\mathcal{C}_k}\right)^{1/3} N_k. \tag{6.17}$$

And the fraction of discs on a given branch i is,

$$f_i = \frac{N_i}{N} = \frac{(\mathcal{C}_i)^{1/3}}{(\mathcal{C}_1)^{1/3} + (\mathcal{C}_2)^{1/3} + ..(\mathcal{C}_k)^{1/3}}. \tag{6.18}$$

For a triangle, which is always composed of three Case (1) branches, the fraction of the discs on a given path can be solved analytically to be

$$f_i = \frac{\cot(\theta_i)}{\cot(\theta_1) + \cot(\theta_2) + \cot(\theta_3)} \tag{6.19}$$

where θ_i is an internal angle of the triangle, each of which is associated with a branch. From equation 6.19, it is clear that the optimal solution preferentially populates medial axis branches associated with smaller internal angles.

In Figure 6.12, the HA of Chapter 6.2.2 was used to optimally fill an irregular arbitrary triangle. On the right of Figure 6.12 the fractional distribution of N disc centers over the triangles pieces, for N = 1-100, is compared to the prediction of Eqn. 6.19. At $N > 20$, the prediction based on the continuum solution is a very good approximation. Even at $N < 20$, the predicted distribution is still a fairly good estimate.

In Figure 6.13, on the left, a log-log plot of the unfilled area of the triangle is shown as N increases. On the right the slope of the log is shown. In both the left and the right, the HA solution (blue) appears to approach a slope of -2 initially but then diverges at $N \approx 45$. A more "constrained" version of the HA was then applied to the triangle, where the relational spacing d between centers on a piece was fixed as per $1/d =\propto r^{-1/3}$. The "constrained" solution, with fewer parameters to optimize, is converging to a slope of -2 until $N \approx 75$ and then also diverges. The divergence appears to be caused by the accumulation of numerical error in the placement of the disc centers by the optimization scheme of the HA. However, the numerical evidence of the optimal filling of this triangle is consistent with the continuum predictions

derived from considering $N \to \infty$.

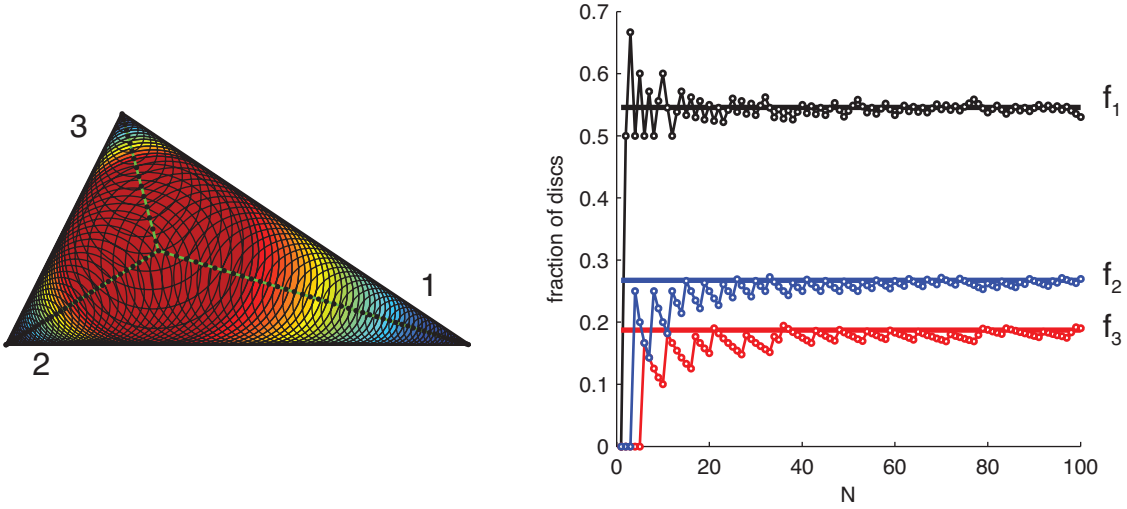


Figure 6.12: The triangle on the left is filled with 100 discs. On the right is the fraction of the discs on each branch for N = 1-100, compared to the prediction per equation 6.19

An illuminating case to consider is a branch of $M(G)$ where both the curvature of the path and the radius function are constant. In Appendix A-3, we find, $C = \frac{1}{12}\left(\kappa^2 r + \frac{1}{r}\right)$, or a constant over the branch. It immediately follows that $\rho = N/T$, where T is the length of the branch. As expected, centers are distributed evenly over the branch. For this case, $\mathcal{C} = T^3 C$. Thus, via equation A.33 $M(G)$ that can be divided into branches of constant curvature and radius functions also have known distributions as $N \to \infty$, and branches with higher curvatures will be more densely populated with disc centers.

6.4 Conclusion

In this chapter we investigated the mathematical structure of the solution of a two-dimensional filling problem. By restricting the problem to simple polygons, the medial axis structure falls into one of three cases.

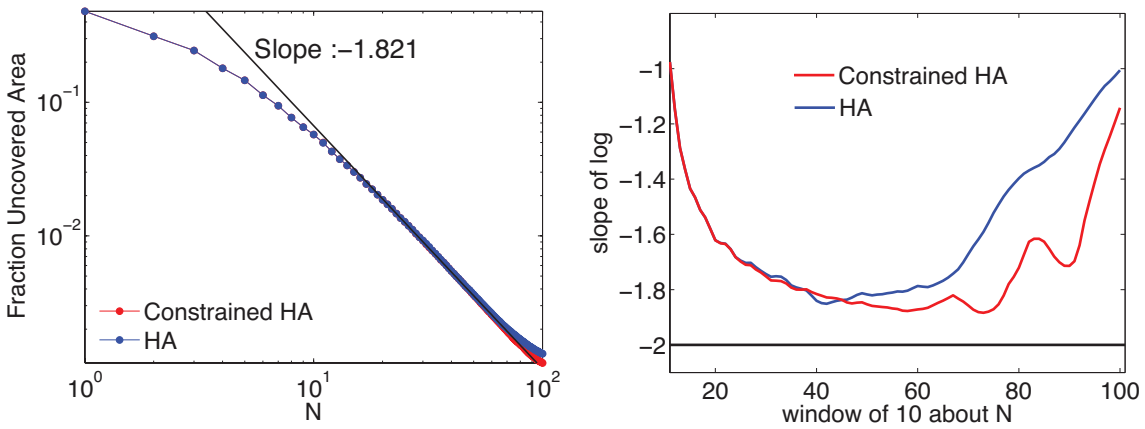


Figure 6.13: On the left is a log-log plot of the unfilled area of the triangle of Fig. 6.19 converging to zero as N increases. On the right, the slope of the log-log plot is shown.

We then introduced two methods by which the optimal filling solutions can be determined: a genetic algorithm that uses almost no assumptions about the structure of the solution, and a heuristic algorithm that exploits the mathematical structure.

We then solved for how infinitely many discs optimally fill a polygon, or the continuum solution. The continuum solution allows optimal fillings to be estimated as $N \to \infty$. The continuum solution also demonstrates the scale invariance of optimal fillings in the corner of a polygon.

In the deceptively simple problem of determining the optimal set of discs to fill an arbitrary planar shape, we have found a surprisingly rich problem, with many open questions, such as how to find optimal solutions for a generalized shape G, or whether for some N, $N > N'$ junction points are always occupied in optimal solutions. By understanding this problem for simple polygons, we found a solution space structure, namely first-order continuous manifolds that join at lower dimension manifolds where centers are trapped, that we expect to find in more generalized and higher dimensional shapes. We predict, for example, that higher dimensional polyhedron shapes will also have manifolds with scale-free solutions. Although originally posed as a method to model anisotropic nanoparticles, filling solutions may have many possible applications

in studies of foams, electromagnetic wave shaping, ablation, and sensor placement.

6.5 Filling Problem Glossary

maximal ball A ball contained completely in a shape G, that is not a proper subset of any other ball also contained in G. Also, a ball tangent to the surface of G at least two points, that completely contained in G. In a 2D planar shape, a maximal ball is a maximal disc.

medial axis $M(G)$, The locus of the centers of all maximal balls of G.

radius function The radii of the maximal balls of a shape G.

normal point A point on $M(G)$ that is the center of a maximal disc in contact with the boundary S at exactly two separate but contiguous sets of points.

end point A point on $M(G)$ that is the center of a maximal disc in contact with the boundary S at exactly one contiguous sets of points.

branch point A point on $M(G)$ that is the center of a maximal disc in contact with the boundary S at three or more separate but contiguous sets of points.

branch A set of contiguous normal point in a medial axis.

parent of a branch the two contiguous parts of S from which the normal points of the branch are derived. For a simple polygon, parents can be a polygon edge or a reflex point.

neighbor If a maximal disc has a disc center that can be reached by a path along $M(G)$ starting at the center of maximal disc A without traversing a third disc center, then it is the neighbor of maximal disc A.

center trap A point on $M(G)$ where a first-order discontinuity coupled with a local maximum in ϕ (all centers fixed) creates a local maximum that is stationary with respect to small changes in the position of the neighboring discs.

junction point A point on $M(G)$ that can act as a center trap. For a polygons, branch points are junction points. A generalized planar graph can have junction points that are not branch points.

piece A junction point or a section of a branch.

way A distribution of N discs over the K pieces that compose $M(G)$.

part A connected set of pieces only connected to pieces not of the set by disc-center occupied junction points.

CHAPTER 7

Isosymmetric Filling Solutions for Platonic Solids and Hypercone Filling

The results of this chapter will be published in:
Phillips, C.L., Anderson, J.A., Chen, E., Glotzer, S.C., Magic Number Filling Solutions for Platonic Solids, Preprint
Phillips, C.L, Chen E., Glotzer, S.C., Scale invariance in an n-dimensional cone fillings for n=2-8 as N→ ∞, preprint

It is unclear if the heuristic algorithm presented in the previous chapter for finding optimal filling solutions of a two-dimensional polygon could be, in principle, extended to a three-dimensional polyhedron. In three dimensions, a medial axis is composed of sheets, seams, and junctions[128–130]. We could divide the sheets, seams, and junctions into pieces, analogous to the junction point and branch pieces of Chapter 6.2.2, but will Conjecture 6.2.1, that distributions of balls over these pieces can have at most one local maximum, still hold?

On a more pragmatic level, while algorithms have been proposed[129,130], software that can produce the medial axis of a polyhedron is not readily available, let alone a straight-forward way to parameterize the cut curved surfaces that would result.

Instead in three dimensions, we rely on a genetic algorithm or simulated annealing algorithm to find local maxima of convex polyhedra. For a restricted number of polyhedra, such as the Platonic solids, where the medial axis structure is immediately evident, we improve on the solutions of the genetic algorithm or simulated annealing

by exploiting the known geometric details of the solution space.

In this chapter we will first describe the algorithms used to find solutions and then discuss the properties of the solutions found for the Platonic solids.

7.1 Platonic Solids

The Platonic solids are convex regular polyhedra, each with faces of only one type of single regular polygon and with the same number of faces meeting at each vertex. The five (and only five) Platonic solids are the tetrahedron, cube, octahedron, icosahedron, and dodecahedron. The Platonic solids are highly symmetrical. The tetrahedron, which is self-dual, has the point group T. The cube and icosahedron, which are duals, have the point group O. The icosahedron and the dodecahedron, which are duals, have the point group I.

The medial axis structure of a Platonic solid naturally reflects the symmetry of the polyhedron. The medial axis contains one junction, the center of the Platonic solid. It has as many seams as vertices, each defined as the segment connecting the vertex to the center and it has many sheets as edges, each defined by the two vertices of the edge and the center point. Each seam of a Platonic solid is symmetrically identical to every other seam and as each sheet is symmetrically identical to every other sheet.

The high symmetry of the medial axis of a Platonic solid provides two benefits. The first is that the medial axis structure can be immediately known. The second is that it is natural to suspect that for each Platonic solid there will be isosymmetric solutions where the filling solution has the same symmetry as that of the polyhedron.

7.2 Algorithm

The filling solutions of the 3D Platonic solids were obtained as follows.

A genetic algorithm or simulated annealing was used to coarsely place balls in the polyhedron. This step can be considered analogous to generating a *way* from Chapter 6.2.2, in that the result is a distribution of centers of balls of junctions, seams, and sheets. This coarse step tends to avoid shallow local maxima, but may also miss deep local maxima with small basins of attraction.

Taking advantage of our explicit knowledge of the medial axis structure, each coarse solution is then maximized as follows. (1) Each ball center is mapped to the closest junction, seam, or sheet. It is assumed that each mapped center is now "trapped" on that manifold. (2) A steepest gradient method is now used to find the local maximum. The gradient of each center is approximated by a central difference calculation only along the manifold the center is mapped to. That is, a center in a junction does not move, a center along a seam can slide along the seam, a center on a sheet can move along the 2-manifold of the sheet. In any other direction off the manifold, the radius function has a first-order discontinuity and the gradient should not be calculated. When the steepest gradient has converged such that the changes in the filling are below a threshold, we assume a local maximum has been found.

For each N a population of 20-60 local maxima were generated via this method. An additional set of maxima were generated by creating "symmetrical solutions". Symmetrical solutions were generatedd by creating a pool of distributions of centers on junctions, seams, and sheets, and then creating a new coarse solution by mapping a single seam distribution to all the seams, a single sheet solution to all the sheets, occupying or not occupying the junction. Using the steepest descent method, the local maxima of each new symmetrical combinations is found.

To calculate the objective function, the volume of the union of the balls, the software *Vorlume*[137,138] was used. *Vorlume* calculates the volume of the union of balls using a certified algorithm, that is, the solution is provided as an interval and the exact volume belongs to the interval computed, so the numerical error in the

calculation is also known. The computational time of this algorithm is linear relative to the number of balls in the system under typical use conditions. The algorithm decomposes the volume into convex regions and returns an interval. The center of the interval was used as the ball volume. The algorithm offers three levels of precision, with increasing computational time per precision level[137]. We tracked the size of the interval of the volume, and switched to a higher level of precision if the interval was too larger ($> 10^{-13}$). All cases where the final interval was larger than 10^{-10} were recorded and if the interval exceeded 10^{-6}, the calculation was terminated and the solution discarded. The latter cases were found to be extremely rare.

7.3 Results

We define a isosymmetric solution as a filling solution that has the same symmetry as the Platonic solid. In figure 7.1, we show the first eight putatively optimal filling solutions for the tetrahedron. Dashed boxes indicate the isosymmetric filling solutions. Non-isosymmtric solutions usually appear as an incomplete or a subset of a particular isosymmetric number filling solution. The $N = 3$ solution, for example, is the N=5 solution minus two small balls.

Given a Platonic solid with E edges and V vertices, isosymmetric filling solutions can only occur when,

$$N = J + V \cdot S_m + E \cdot S_t \tag{7.1}$$

where N is the number of balls in the solution, $J = \{0, 1\}$ (i.e. whether the center is occupied), S_m the number of ball centers on a single seam, and S_t the number of ball centers on a single sheet.

In this section we show the isosymmetric solutions that were found for each Platonic solid. While we cannot rule out that at each N shown, a non-isosymmetric solution may be the true global maximum, we can affirm that each solution shown is the

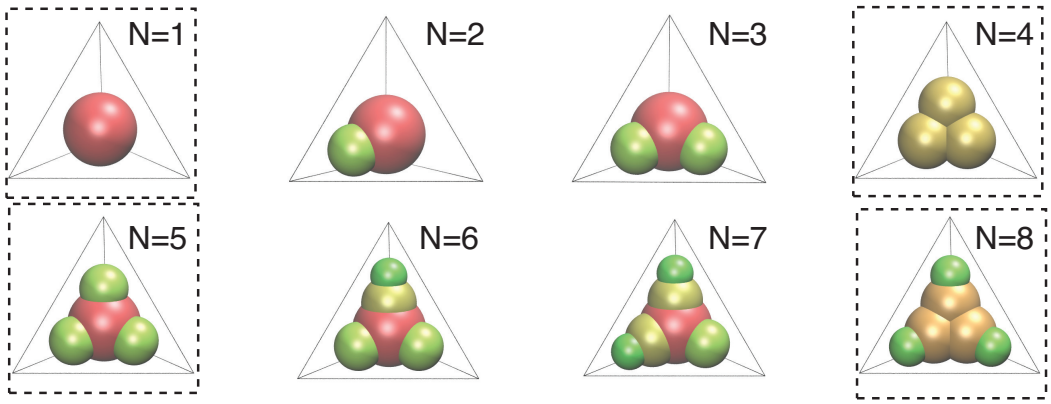


Figure 7.1: The N=1-8 putative global filling solutions are shown for the tetrahedron. The isosymmetric solutions are in dashed boxes.

largest maximum we found. For each Platonic solid, the balls filling the polyhedron are colored on a red-green-blue spectrum so that the largest ball of the polyhedron, or the insphere, is colored red and the smallest (theoretically of radius zero) is dark blue[75]. Each solution is labeled with its N and the (J, S_m, S_t) of the solution. The solutions for the tetrahedron, cube, octahedron, icosahedron, and dodecahedron are shown in Figures 7.2, 7.3, 7.4, 7.5, and 7.6, respectively.

One observation we make is that isosymmetric solutions do not exist for every triple of natural numbers (J, S_m, S_t). When N is small, isosymmetric solutions that one might expect to be present are not. For example, only the tetrahedron has (0,1,0) as a isosymmetric solution. As N becomes large, we observe that the distribution of the centers on a junction, two seams, and the triangular sheet between them, forms a self-similar triangular-like structure, and thus the number of ball centers on the combined structure is near a triangular number, or $(J, S_m, S_t) = (1,\ n, K)$, where $K \approx T_n - (2n-1)$, $T_n = \binom{n+1}{2}$.

A surprising result is that, for the cube, the N=8 optimal filling solution is neither a isosymmetric solution, nor is it simply the N=9 solution minus a single small ball. The N=8 optimal filling solution has tetrahedral symmetry instead of octahedral symmetry. In Figure 7.7 we compare the best octahedral symmetry N=8 filling

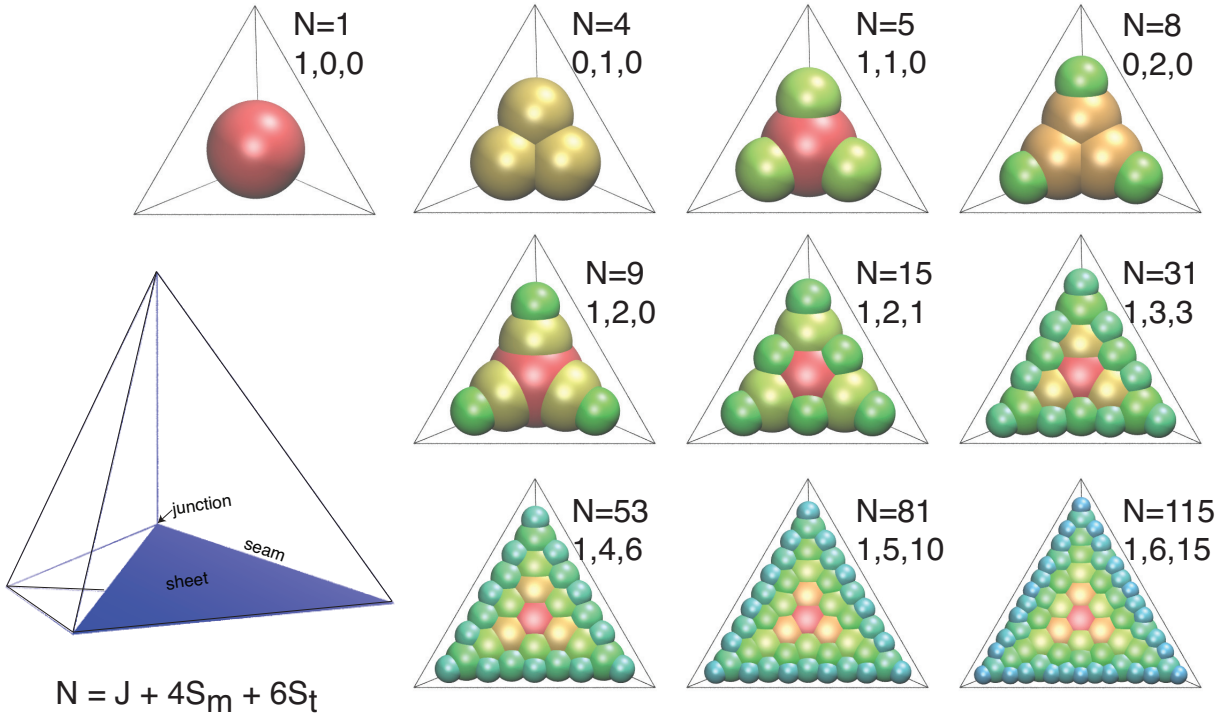


Figure 7.2: For a regular tetrahedron, the medial axis consists of six triangular sheets that join at four seams and at a single junction equal to the center of the tetrahedron. Each isosymmetric configuration is labeled with its (J, S_m, S_t).

solution to the best tetrahedral symmetry solution. The color scheme in Figure 7.7 is renormalized so that the two ball sizes of the tetrahedrally symmetric solution are the limits of the color spectrum to make the difference in ball radii more clear. On the left, the two best solutions are overlaid on each other. On the right side the filling value ϕ of all the octahedrally symmetric N=8 solutions of the form (0,1,0) are generated as a function of their offset from the center along a unit-length seam. At an offset of zero, the 8 balls cover the same volume as the in-sphere and $\phi = \pi/6$. At an offset of 1.0, the 8 balls have radius zero and $\phi = 0$. We observe that the difference between the best octahedral and tetrahedral solution is $\Delta\phi \approx 10^{-5}$. On closer examination, we find that the best octahedral symmetry point is a saddle-point.

We also observe that different Platonic solids demonstrate different trends with regard to occupying or not occupying the junction at the center of the shape. The

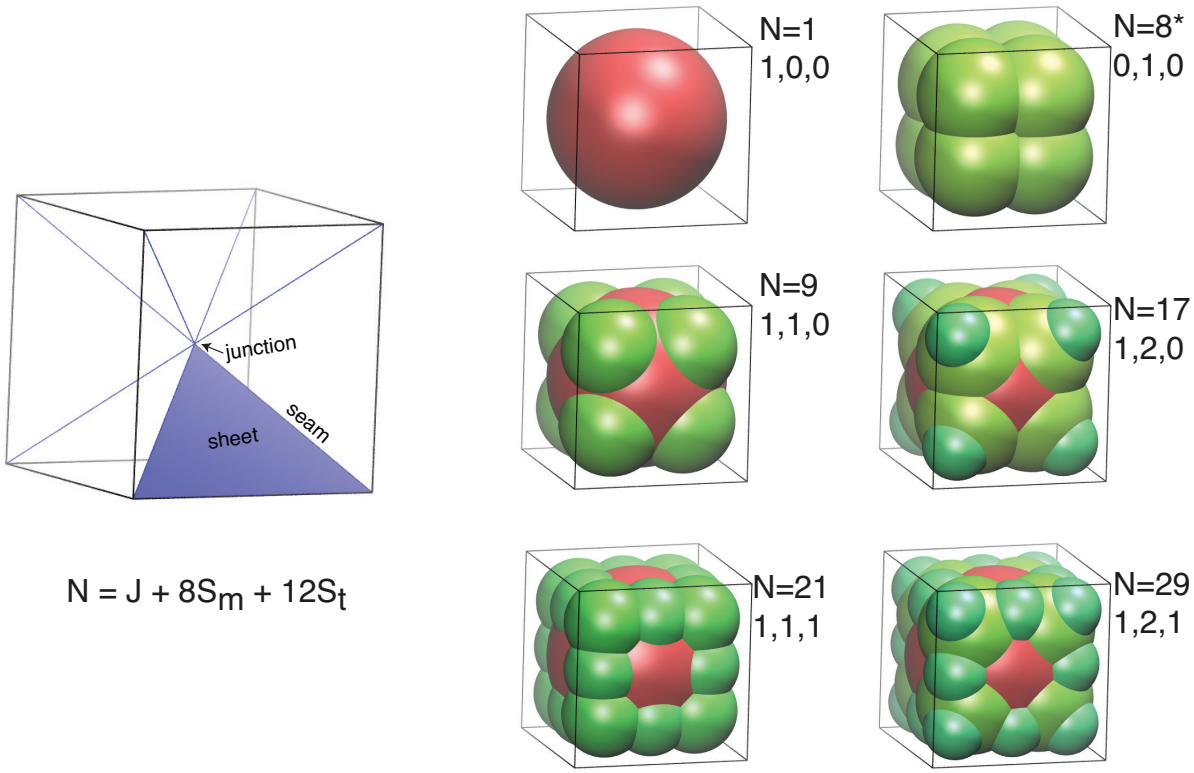


Figure 7.3: For a regular cube, the medial axis consists of twelve triangular sheets that join at eight seams and at a single junction equal to the center of the cube. Each isosymmetric configuration is labeled with its (J, S_m, S_t). *The N=8 case is not a isosymmetric and is discussed in more detail.

icosahedron, for example, seems to show a strong preference for maxima without occupied junctions. We see no simple logic to explain this, given that the dodecahedron does not show this preference.

7.4 Scaling and Convergence

In chapter 6.3, we were able to make considerable progress in understanding how discs ideally pack in different arrangements found in polygons by finding the lowest order terms of the uncovered area between overlapping discs as $N \to \infty$. Numerical data at finite N supported the continuum predictions. However the numerical evidence showed both that a sufficient number of discs were needed before numeri-

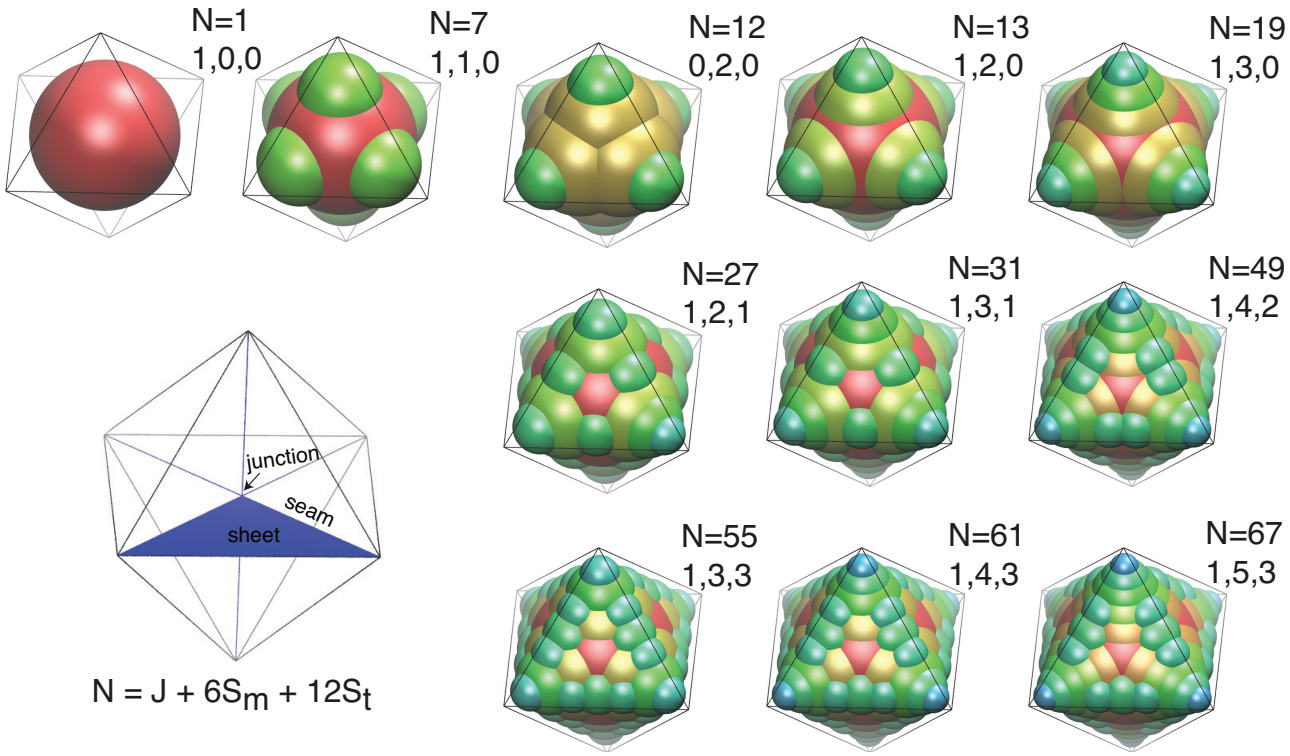


Figure 7.4: For a regular octahedron, the medial axis consists of twelve triangular sheets that join at six seams and at a single junction equal to the center of the octahedron. Each isosymmetric configuration is labeled with its (J, S_m, S_t).

cal data approached continuum predictions, and that as N increased, the numerical data accumulated errors causing it to diverge from continuum predictions. In three-dimensions, it is reasonable to expect both that even higher N is required to converge to continuum solutions and that the method used to find solutions will be prone to accumulating numerical error.

In Figure 7.8, we show the convergence towards zero of the unfilled volume of the tetrahedron from Fig. 7.2 as N increases. As one would intuitively expect, the convergence of the filling in a tetrahedron appears slower ($\approx N^{-0.53}$) than in a polygon ($\approx N^{-2}$), and the numerical evidence is insufficient to indicate at what rate the filling converges to as $N \to \infty$. First, N is likely too low for the high order terms to vanish. Second, as N becomes high, the numerical method used will accumulate more errors in finding ideal coordinates for each ball.

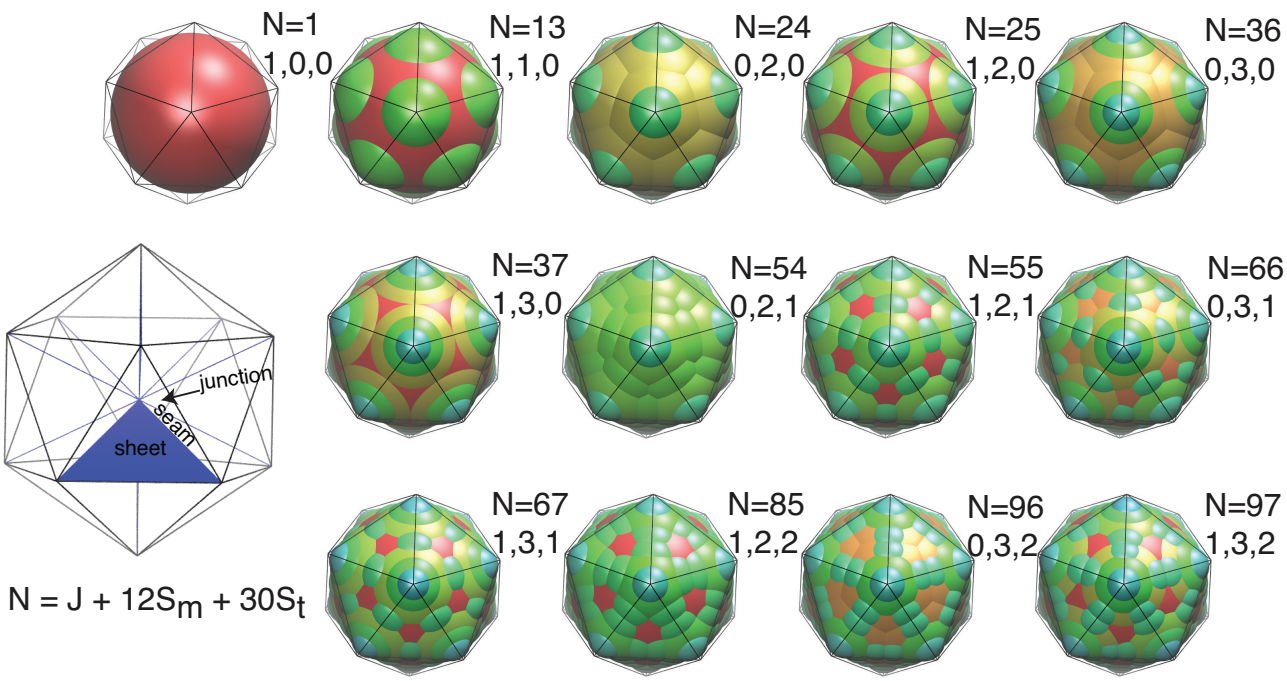


Figure 7.5: For a regular icosahedron, the medial axis consists of thirty triangular sheets that join at twelve seams and at a single junction equal to the center of the icosahedron. Each isosymmetric configuration is labeled with its (J, S_m, S_t).

In the next two sections we consider a few very simple geometries where we can derive both continuum convergence laws and scaling laws in lower and higher dimensions.

7.4.1 Filling between two infinite parallel planes

In this subsection we consider the question of the convergence of a filling of balls between two infinite parallel planes. Assume you have two infinite parallel planes of distance two units apart. The medial axis of the two parallel plane is an infinite parallel planar sheet halfway between the planes with a radius function of one unit.

To fill the space between the planes with balls, we fill the space with a hexagonal lattice of unit radii balls with a lattice spacing of d. The uncovered volume is the wedges between three neighboring balls on an equilateral triangle. We approximate this volume as an irregular tetrahedron whose base is the equilateral triangle of contact

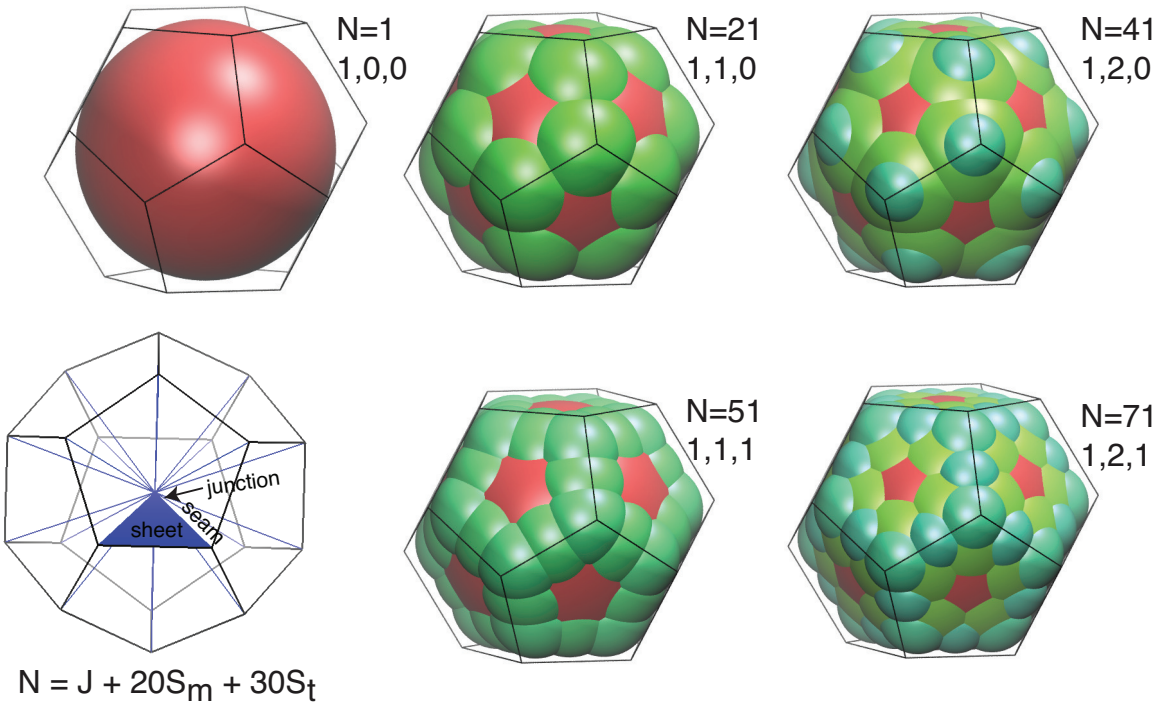


Figure 7.6: For a regular dodecahedron, the medial axis consists of thirty triangular sheets that join at twenty seams and at a single junction equal to the center of the dodecahedron. Each isosymmetric configuration is labeled with its (J, S_m, S_t).

points of the three balls with the plane and whose fourth vertex is the point of intersection of the three balls closest to the plane. The height of the fourth vertex is $h = 1 - \sqrt{1 - d^2/3}$, so $V_t = \frac{\sqrt{3}d^2}{4}(1 - \sqrt{1 - d^2/3})$ For small d, using a Taylor expansion, we find $V_t \approx \frac{d^4}{12\sqrt{3}}$. Assume there are N balls per unit area. The density of these tetrahedra is 2N and $d \propto \sqrt{\frac{1}{N}}$. So summing all the tetrahedra, the total uncovered volume in the unit area is proportional to $N/(N^{1/2})^4 = 1/N$. This implies that the filled volume converges to the total volume with an error proportional to $1/N$ as $N \to \infty$ between two parallel planes.

To test this approximation numerically, a triangular lattice of N lattice points was constructed within an equilateral triangle of side 1.0. Balls of radius 1 were placed on the lattice points. The volume of the puffy triangle being filled is the volume of the triangular plate, three half cylinders, and a sphere, or $\sqrt{3}/2 + 3\pi/2 + 4\pi/3$. The

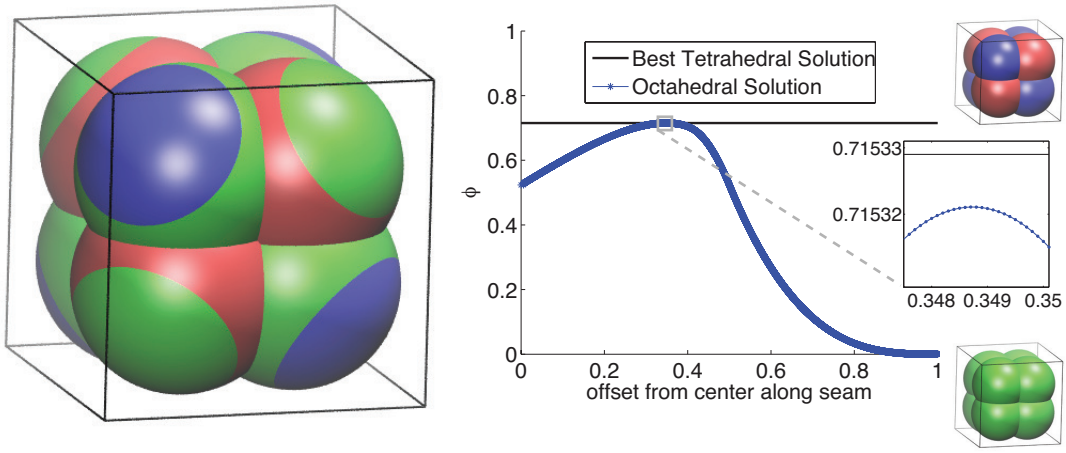


Figure 7.7: On the left, the best tetrahedral solution and best octahedral solution are overlaid on each other. In the plot on the right, the ϕ of the best tetrahedral solution is compared to the ϕ of all octahedral solutions. As can be seen in the inset, the best tetrahedral solution fills the cube by $\Delta\phi = 10^{-5}$ more than the best octahedral solution.

log-log plot of the uncovered volume as a function of N and the numerical evidence of the convergence of the slope to -1 is shown in Figure 7.9.

This filling scheme between two planes is trivially scale-invariant. The appearance of the distribution is invariant if the density of centers is increases in linear proportion to the unit area.

Having considered the case of the infinite parallel planes, we naturally would like to understand how balls will first, fill, and second, converge between two planes that intersect at an edge. By examining the N=115 tetrahedron, we suspect that a local hexagonal motif will still be present. Will the height of the hexagonal lattice simply diminish according to a power law as the filling approaches the edge? Will additional balls be inserted into the lattice like an infinite binary tree or a hyperbolic fractal as the filling approaches the edge? We don't know. This is an open question of 3D filling.

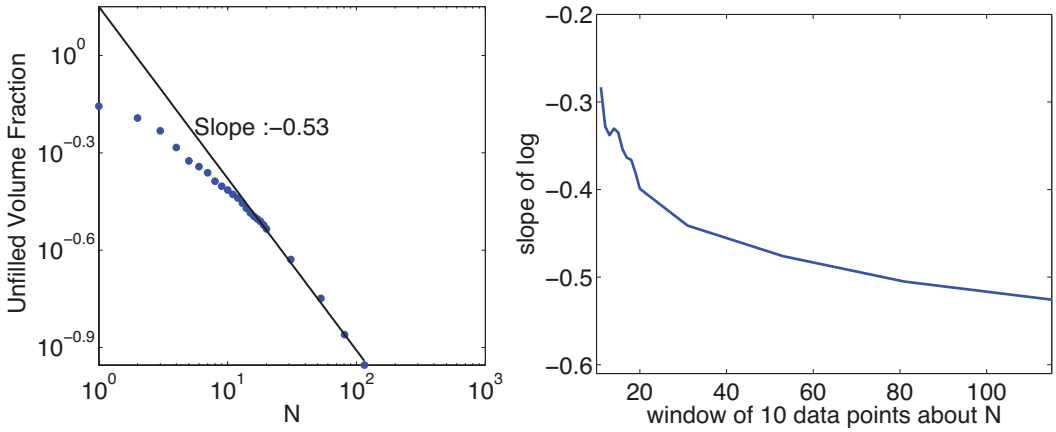


Figure 7.8: On the left is a log-log plot of the unfilled area of the tetrahedron of Fig. 7.2 converging to zero as N increases. On the right, the slope of the log-log plot is shown.

7.4.2 Filling in a cone in two to eight dimensions

Consider a n-dimensional infinite cone filled with an infinite number of n-balls. The unfilled volume between two n-balls of radius r and s whose centers are distance d apart can be expressed as and inclusion-exclusion formula (Figure 7.10 and Eqn. 7.7) as derived in Appendix A.6,

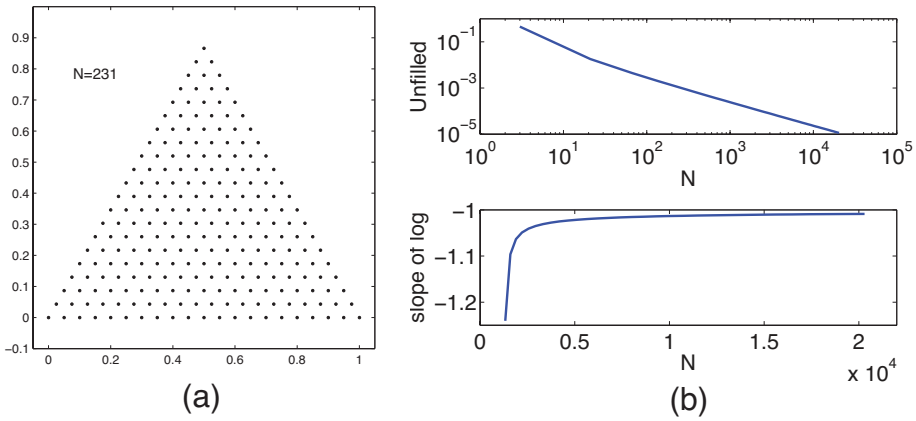


Figure 7.9: (a) The arrangement of the centers of unit balls inside a triangle. (b) The unfilled volume of the triangle (top) and the convergence of the slope to -1 (bottom) as a function of N balls.

$$
\begin{aligned}
V &= \frac{\pi^{\frac{n-1}{2}}}{\left(\frac{n-1}{2}\right)!}\frac{d}{n}\frac{s^n - r^n}{s-r}\cos\left(\theta/2\right)^{n-1} && (7.2)\\
&\quad -\frac{\pi^{\frac{n}{2}}}{\left(\frac{n}{2}\right)!}r^n && (7.3)\\
&\quad -\frac{\pi^{\frac{n-1}{2}}}{\left(\frac{n-1}{2}\right)!}(s^n - r^n)\frac{n-1}{n}\int_0^{\frac{\pi}{2}-\theta/2}\sin(a)^{n-2}da && (7.4)\\
&\quad +\frac{\pi^{\frac{n-1}{2}}}{\left(\frac{n-1}{2}\right)!}s^n\frac{n-1}{n}\int_0^{\mathrm{ArcCos}\left(\frac{d^2+s^2-r^2}{2sd}\right)}\sin(a)^{n-2}da && (7.5)\\
&\quad +\frac{\pi^{\frac{n-1}{2}}}{\left(\frac{n-1}{2}\right)!}r^n\frac{n-1}{n}\int_0^{\mathrm{ArcCos}\left(\frac{d^2+r^2-s^2}{2rd}\right)}\sin(a)^{n-2}da && (7.6)\\
&\quad \frac{\pi^{\frac{n-1}{2}}}{\left(\frac{n-1}{2}\right)!}\frac{d}{n}\frac{\cos\left(\theta/2\right)^{n-1}}{2^{n-1}}\left((s+r)^2 - d^2\right)^{\frac{n-1}{2}} && (7.7)
\end{aligned}
$$

The lowest order terms of d of this equation found for dimension n = 2,3,4,5,6,7,

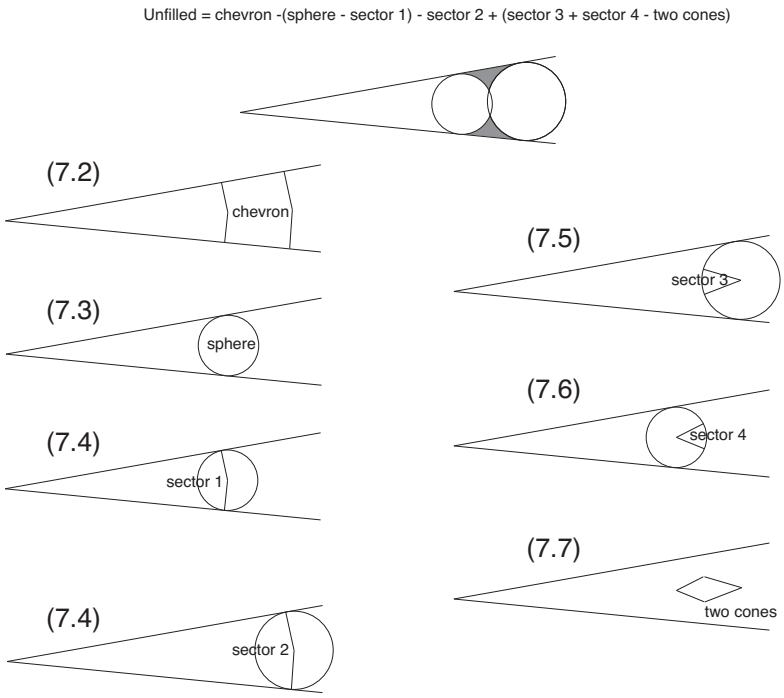


Figure 7.10: (a) The geometric pieces of the inclusion-exclusion formula of Eqn. 7.7.

and 8 are as follows:

$$n = 2; \quad V_2 \approx \frac{\cos(\theta/2)^3}{12}\frac{d^3}{r} = C_2 d^3 \tag{7.8}$$

$$n = 3; \quad V_3 \approx \frac{\pi\cos(\theta/2)^4}{12} d^3 = C_3 d^3 \tag{7.9}$$

$$n = 4; \quad V_4 \approx \frac{\pi\cos(\theta/2)^5}{6} r d^3 = C_4 d^3 \tag{7.10}$$

$$n = 5; \quad V_5 \approx \frac{\pi^2\cos(\theta/2)^6}{12} r^2 d^3 = C_5 d^3 \tag{7.11}$$

$$n = 6; \quad V_6 \approx \frac{\pi^2\cos(\theta/2)^7}{9} r^3 d^3 = C_6 d^3 \tag{7.12}$$

$$n = 7; \quad V_7 \approx \frac{\pi^3\cos(\theta/2)^8}{24} r^4 d^3 = C_7 d^3 \tag{7.13}$$

$$n = 8; \quad V_8 \approx \frac{2\pi^3\cos(\theta/2)^9}{45} r^5 d^3 = C_8 d^3 \tag{7.14}$$

As in Chapter 6.3, to solve for the ideal density distribution, ρ of n-ball centers along the one-dimensional hypercone medial axis as the number of balls, $N \to \infty$, we sum the unfilled volumes along the length of the hypercone ($d = 1/\rho$, the density of

volumes locally is ρ).

$$V_{uncovered} = \int_{t_b}^{t_a} V_n \rho dt = \int_{t_b}^{t_a} C_n \frac{1}{\rho^2} dt \tag{7.15}$$

As this equation fits the form of 6.2, in the continuum limit the filling converges to the volume of the hypercone with an asymptotic error proportional to N^{-2} for ideally distributed centers

The form of ρ is known per 6.7,

$$\rho \propto C_n^{1/3}. \tag{7.16}$$

We now can determine the dependence of ρ on the only part of C_n that changes along the medial axis of the hypercone, the radius.

$$n = 2; \quad \rho \propto r^{-1/3} \tag{7.17}$$

$$n = 3; \quad \rho \propto 1 \tag{7.18}$$

$$n = 4; \quad \rho \propto r^{1/3} \tag{7.19}$$

$$n = 5; \quad \rho \propto r^{2/3} \tag{7.20}$$

$$n = 6; \quad \rho \propto r \tag{7.21}$$

$$n = 7; \quad \rho \propto r^{4/3} \tag{7.22}$$

$$n = 8; \quad \rho \propto r^{5/3}. \tag{7.23}$$

The relationship of ρ relative to the radius function changes with increasing dimension. While in a two-dimensional cone (two intersecting lines), discs are found more densely at smaller the smaller radii, in a three-dimensional cone, centers are *evenly* distributed along the entire length of the cone. In higher dimensions, centers

are more densely distributed at the larger end of the hypercone.

Also, notably, the convergence to the shape in a cone versus between two parallel planes in three dimensions is different (N^{-2} vs. N^{-1}).

The relationship of scale-free packing of n-balls in a hyper-cone can be acquired from

$$N = \int_0^T \rho(t)dt = \rho_0 \int_0^T r^{(n-3)/3}dt \tag{7.24}$$

where T is the height of the cone. In the cone, $r(t) = t\sin(\theta/2)$, thus

$$N = \rho_0(3/n)\tan(\sin/2)^{(n-3)/3}T^{n/3} \tag{7.25}$$

or if the height of the cone is scaled by λ, e.g. $T' \to \lambda T$, and the number of centers is scaled by $\lambda^{n/3}$, e.g $N' = \lambda^{n/3}N$, or the scaling dimension is $-n/3$.

For comparison, consider the scaling law that would have resulted from packing N hard non-overlapping n-balls in a cone with centers on the medial axis, where $d = r + r + \Delta r$, or $d = At$ for $A = 2(\sin(\theta/2))/(1 - \sin(\theta/2)$. In this case,

$$N = \int_{T_b}^{T_a} \rho(t)dt = \int_{T_b}^{T_a} t^{-1}dt = A^{-1} \cdot \ln(T_a/T_b) = A^{-1} \cdot \ln(1 + T/T_b). \tag{7.26}$$

We see that if $T \to \lambda T$ and $T_a \to \lambda T_a$, N is the same, or $N \to \lambda^{-0}N$. The packing of hard n-balls in a hypercone is naturally scale invariant, but unlike filling, the scaling dimension is independent of the dimension n.

7.5 Conclusion

In this chapter we show the first optimal filling solutions of 3D polyhedra. We focus on the filling solutions of the Platonic solids, first because their medial axis can be readily identified, and second, because their solutions are expected to evidence high

symmetry at for certain optimal fillings, or at isosymmetric solutions. We identify ten, five, eleven, thirteen, and five isosymmetric solutions for the tetrahedron, cube, octahedron, icosahedron, and dodecahedron.

Magic numbers higher than what we found are expected, however, our method is computationally limited due to both the impressive but non-negligible precision of our objective function calculation and the high computational cost of high N solutions.

The methods demonstrated in this chapter could be applied to generalized polyhedra, if the medial axis structure could be identified.

We also consider the filling scaling and convergence between two infinite two-dimensional planes and hypercones from dimension two to seven. We find that in the third dimension, balls filling the volume between two infinite planes and in a hypercube converge to the volume at different rates and follow different scaling laws. We find that from dimension two to seven, all hypercones converge at the same rate, but that the distribution as a function of the radius function changes. At dimension four and higher, ball centers are found more densely at the larger end of a hypercone than the smaller.

We see that there are many open questions as to how spaces are filled in higher dimensions. The lattice formed by the centers filling two intersecting planes, for example, is still an open question. We propose that the convergence of the filling between two planes can probably be extended to higher dimensional hyperplanes, and presume that the best $n-1$ dimensional lattice packing fills the volume between the hyperplanes. We see this as an area of mathematics with many fruitful directions and open questions.

CHAPTER 8

Effect of Nanoparticle Polydispersity on the Self Assembly of Polymer Tethered Nanospheres

This chapter corresponds to publications:

Phillips, C.L., Iacovella, C.R., Glotzer, S.C., Stability of the double gyroid phase to nanoparticle polydispersity in polymer-tethered nanosphere systems, Soft Matter, 6, 1693 - 1703, 2010

Phillips C.L., Glotzer S. C., Effect of nanoparticle polydispersity on the self assembly of polymer tethered nanospheres, Preprint

The ability of block copolymers to phase separate into periodic micro-domains makes them attractive building blocks for engineering self-assembled nanomaterials[139–141]. Possible applications of the periodic nanometer-sized domains include microelectronics[142] and high-density storage media[143], photonic band gap materials[144,145], and drug delivery systems[146,147]. Recent attention has focused on the use of polymer-tethered nanoparticles as a means to create novel nano-materials by exploiting the block copolymer-like immiscibility between nanoparticle and tether[5,29,122,148]. Several techniques exist to create composite polymer-nanoparticles. Westenhoff and Kotov, for example, used poly(ethyleneglycole) PEG polymer to tether a CdTe nanoparticle to a surface[149]. Several groups have created gold or SiO_2 nanoparticles functionalized with polymers or DNA linkers[150–152] Even more advanced techniques are being proposed to create nanoparticles with multiple functionalizations with controlled placements for creating self-assembled structures[153,154]. Polymer tethered nanosphere amphiphiles[122] are thus currently realizable.

Iacovella and coworkers[24,25,155] predicted using computer simulations that polymer-tethered nanospheres (NS) (fig. 8.1) form, under suitable conditions, phases similar to block copolymers[156–158]. The ordered phases found when the NS head group is in poor solvent are hexagonally packed cylinders (H), the double gyroid (DG), perforated lamellae (PLH), and lamellar bilayers (L), as shown in Figures 8.2, 8.3, and 8.4. At temperatures outside the stability range of these phases the tethered nanoparticles still aggregrate, but no ordered structure is found. This region is characterized by disordered wormy micelles (DWM).

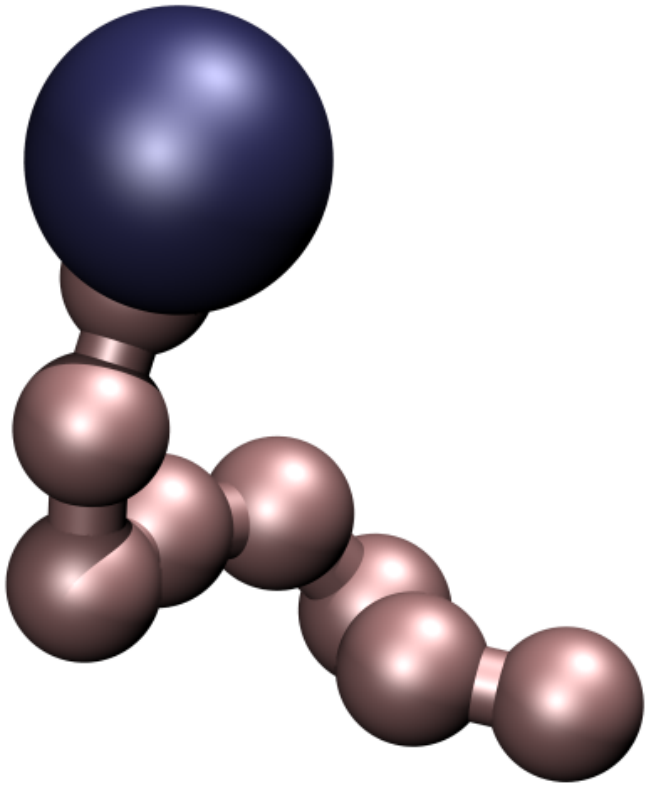


Figure 8.1: A polymer functionalized nanosphere of diameter 2. The polymer is modeled as 8 soft sphere (WCA) beads connected by FENE springs.

The phase diagram predicted by Iacovella et al. was based on monodisperse tethered nanospheres (TNS) with uniform diameter NS. In all nanoparticle synthesis approaches, the polydispersity is non-zero, and in some cases can be appreciа-

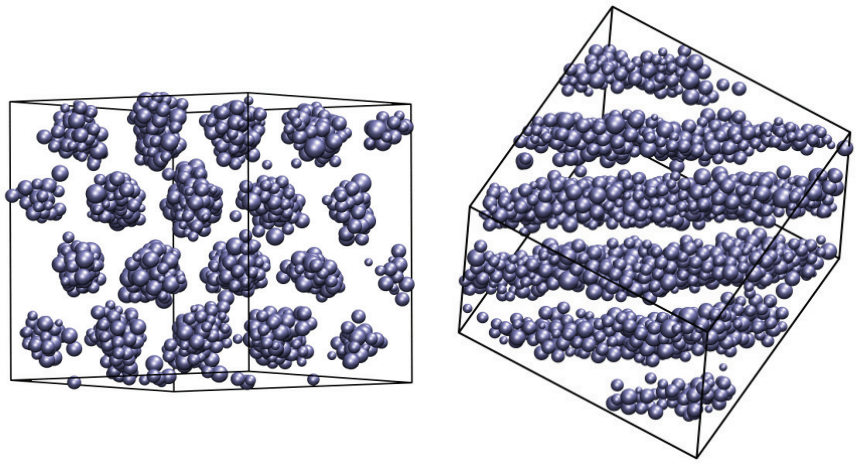


Figure 8.2: Two views of a simulation cell containing 2000 TNS in the H phase. NS are blue; Tethers are not shown. The NS in this system have polydispersity $\Delta = 20\%$.

ble. State-of-the-art techniques are able to achieve nanoparticles with polydispersity values as low as 6%[159–161]. For other self-assembling liquid crystal or hard sphere systems, it has been recognized that certain crystalline orderings can only tolerate a certain level of polydispersity and still be a stable phase; that is, they exhibit terminal polydispersity[162]. For example, Pusey (1987) argued that crystallization of hard sphere colloids would have a terminal polydispersity between 6-11%[163], a range supported by subsequent experiments[164]. Terminal polydispersity for different systems has been studied experimentally[165], analytically[166], and computationally[162,167].

In this chapter, we consider the how polydispersity affects all the ordered phases of the phase diagram of the tethered nanosphere system. We consider, for example, how polydispersity influences the order-disorder temperature T^*_{ODT} between the DWM phase and ordered phases, and whether, beyond a "terminal polydispersity", a different phase may be stabilized. We are also interested in how increasing polydispersity affects internal properties of the phase, such as the local packing structure and packing fraction. Understanding the influence of polydispersity on these properties may explain why phases become stabilized or destabilized by increasing polydispersity,

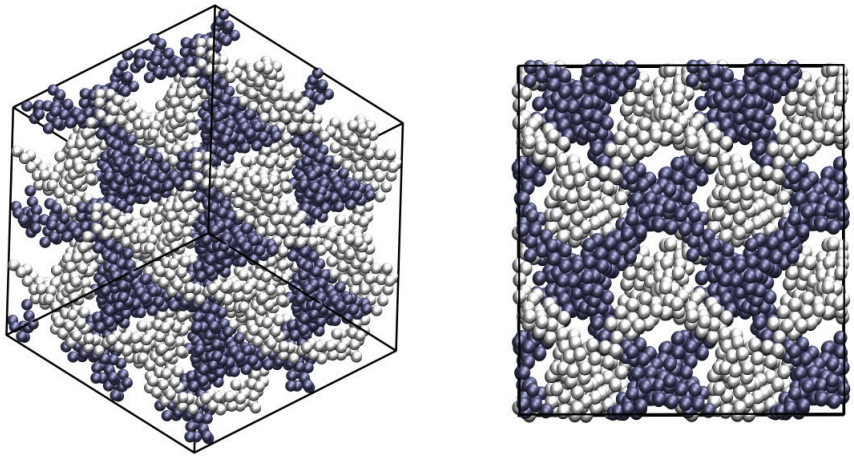


Figure 8.3: The left panel shows 8 unit cells of a double gyroid (DG) phase. The NS are monodisperse and shown in blue and white. Tethers are not shown. The right panel shows the same 8 unit cells from a side perspective.

and provide insight into the internal structure of materials created from polydisperse TNS. We pay special attention to the exotic DG phase. The DG is a triply periodic structure of space group Ia3d where space is divided into three regions: two interpenetrating but identical networks (here, the NS domain) and a matrix (polymer tether domain). The surface separating the domains is approximately a surface of constant mean curvature and minimizes the interfacial area subject to a volume constraint [168].

Analysis of phase stability is based on using two separate "paths" through the phase diagram to identify the stable state. For one path the system is initialized and equilibrated at a high temperature to a random disordered state. The simulation is then cooled and reequilibrated at the new temperature. Ordered phases that form at a given temperature via different cooling schedules represent free energy minima. This "path" through the phase diagram resembles actual self-assembly of a physical system, albeit using a cooling schedule that is likely accelerated by several orders of magnitude relative to a physical system. This path shall be subsequently referred to as the Conventional Path (CP). The second path considers polydispersity as a perturbation to the monodisperse state. Polydispersity is introduced to a monodisperse ordered

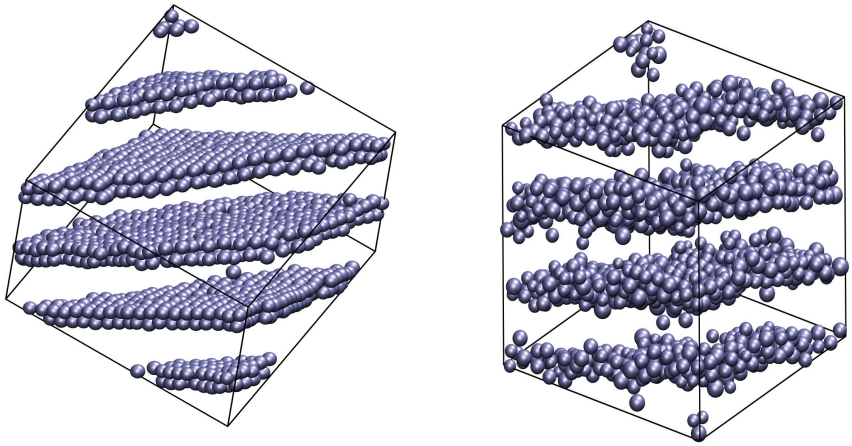


Figure 8.4: The left panel shows 2000 TNS that have self-assembled to the L phase by a Conventional Path. The NS in this system have polydispersity $\Delta = 3\%$. The right panel shows 2000 TNS that have self-assembled to the PLH phase by a Alternate Path. The NS in this system have polydispersity $\Delta = 10\%$.

phase by "growing" the NS to a target diameter distribution while at low temperature. The system is then heated and equilibrated to determine if the phase remains stable. This path shall be subsequently referred to as the Alternate Path (AP).

In the thermodynamic limit, these two "paths" through the phase diagram should produce identical free energy minima. In practice, however, at state points where both a disordered arrangement and an ordered arrangement may locally minimize the free energy, the CP is biased towards being kinetically trapped in the disordered phase and the AP is biased towards being kinetically trapped in the ordered phase. Further analytical tools would be required to determine which of the two phases minimizes the free energy globally, and which represents a metastable phase. Thus we use these two paths in this Chapter to bound the possible phase diagrams of the TNS system subject to polydispersity.

In section 8.1 of this chapter, we describe the method we use to model polydisperse TNS and how the polydispersity of a system can be changed continuously in a molecular dynamics simulation. In section 8.2 we describe the analytical techniques we use to study the packing properties of polydisperse nanospheres, namely (1) the

use of the structure factor to identify the double gyroid phase, (2) the R_{YLM} structure analysis used to identify the local ordering of particles within the phase, and (3) the Voronoi tessellation, extended to handle polydisperse spheres and used to identify how increasing the polydispersity affects the net packing of nanospheres. In section 8.3 we present the results of our simulations and analysis. In subsection 8.3.1, we consider the CP phase diagram of the DG in detail. In subsection 8.3.2, we compare the phase diagrams generated by CP and AP of all the phases. In subsection 8.3.3, we consider how properties of the system, such as internal structure, packing fraction, potential energy, and coordination number are affected by polydispersity. In section 8.4, we discuss the the various results of the structure analyses to determine how polydispersity affects the local icosahedral packing and analyze why a low level of polydispersity promotes local icosahedral structure. We focus on the DG phase. In section 8.5 we provide concluding remarks.

8.1 Methods

8.1.1 Simulation methods

Simulations are modeled as per the method outlined in 2, using Brownian Dynamic and modeling the TNS as SLJ spheres with eight bead FENE polymer tethers.. For this system, the BD drag force scales with the size of the particle.

A set of polydisperse NS is created by sampling from a Gaussian distribution of particle diameters σ:

$$P(\sigma) = \frac{1}{\delta\sqrt{2\pi}} exp\left[-\frac{1}{2}\left(\frac{\sigma - \bar{\sigma}}{\delta}\right)\right] \tag{8.1}$$

The non-dimensionalized polydispersity, Δ, is defined as $100\delta/\hat{\sigma}$, where δ is the standard deviation and $\hat{\sigma}$ is the average diameter. Normally distributed populations of diameters are generated. The population of diameters is then shifted or scaled so that the net volume of the nanoparticles is kept constant in order to prevent the bulk

system volume fraction from deviating as polydispersity is introduced. The diameter distribution is also truncated at a minimum value of 1.0 so that all the NS are at least as large as a tether bead. The NS are polydisperse in size only. The mass of each nanoparticle is kept fixed at $m_{NS} = 27m_{tether}$ to minimize the number of variables in the system. Since any simulation contains a finite set of NS drawn from this distribution, the nominal polydispersity of the distribution, i.e. the polydispersity of the distribution being sampled, and the actual polydispersity of the set generated differ slightly. In this chapter, the nominal polydispersity of the distribution is presented as an integer value. When the polydispersity has a decimal component, this represents the actual polydispersity of the distribution. Polydispersity values reported on plots are actual Δ values.

8.1.2 Bounding the phase diagram

At a given volume fraction, ϕ, upper and lower bounds can be established for T^*_{ODT} and the terminal value of Δ for ordered phases on a polydispersity vs. temperature phase diagram by using different themodynamic paths. Specifically, we explore the phase diagram by changing the variables of temperature T^* and polydispersity Δ. A diagram of an example of a CP and AP is shown in Figure 8.5. For both paths, simulations are initiated at a high temperature disordered monodisperse NS state. For a Conventional Path, we first slowly adjust the diameters of the NS to a polydisperse distribution. As the change to the diameters is performed while the system is hot, diffusive, and disordered, the state of the simulation when the polydisperse distribution is reached is indistinguishable from a simulation initialized with a polydisperse NS distribution. The simulation is then cooled via a cooling schedule to avoid kinetically trapped states. In studies of self-assembly using molecular dynamics (including BD), using a cooling schedule from a hot disordered state is a standard way the ordered phases of phase diagrams are generated. Cooling a system from a disordered state also

emulates how physical particles self-assemble into an ordered state. Usually, when an ordered state is produced repeatably, by different cooling schedules, or for different simulation sizes, cooling from a disordered state is considered to have produced the free energy minimizing state point. However, for kinetic reasons, or due to sensitivity of a phase to simulation box size, the free energy minimizing phase for a given state point may not be found in simulation. In this chapter we will compare the phase diagram generated by the CP to an alternate thermodynamic path (AP). For the AP, we start with a monodisperse disordered high temperature system at a given ϕ. The temperature is lowered by a cooling schedule to a $T^* < T^*_{ODT}$, so that the system is in the mondisperse ordered phase. In practice, only simulations with a minimal amount of defects in the ordered phase are retained for the next step. Polydispersity of the NS is then slowly increased to the desired Δ. The simulation is then heated via a schedule to a desired T^*.

The AP provides a different thermodynamic path for approaching a state point. For phases that are challenging to self-assemble (e.g. the DG phase), an advantage of the AP over the CP is that many independent simulations of the ordered phase at a given Δ, ϕ, and T^* can be rapidly and reliably generated. The sensitive phase may self-assemble only rarely using a CP method. The disadvantage of the AP is that it provides no information as to kinetic difficulty in forming the ordered phase from a disordered state.

8.1.3 Low temperature response to polydispersity

We introduce a method for calculating the properties of a system as a continuous function of Δ without requiring a large number of computationally-intensive independent CP simulations at fixed values of Δ intervals. To study Δ as a continuous variable, polydispersity is continuously added to the NS extremely slowly, or quasi-statically. The polydispersity is changed slowly enough that the system is

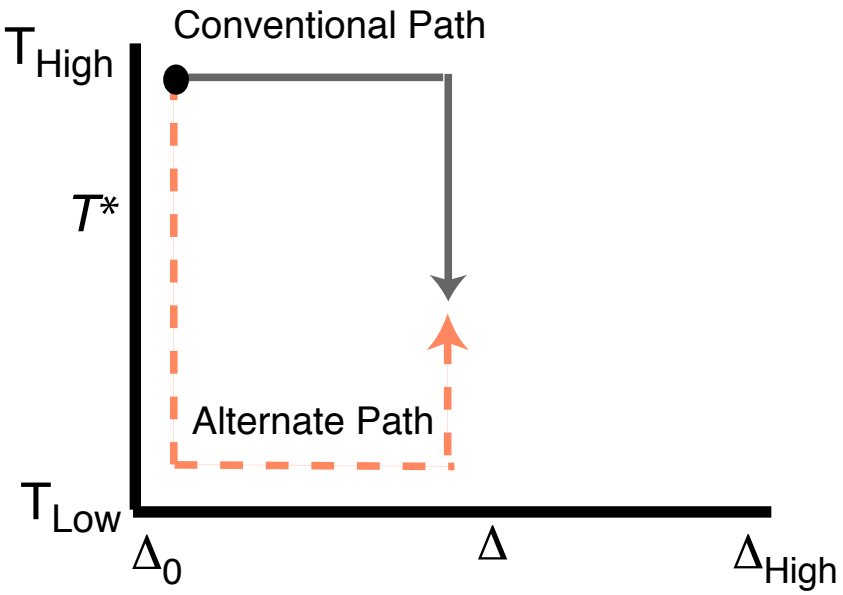


Figure 8.5: Schematic showing the two paths used in the present study

able to relax in response to the change and the properties of the system, (barring kinetically-trapped phase transition regions) calculated as a function of time, closely approximate those of the equilibrated system.

We will use this method to study the H, PLH, DG, and L phase. At $\phi = 0.25$, 0.3, and 0.4, we start with an ordered monodisperse system at $T^* = 0.25$, (i.e. H, DG, and L phase). The polydispersity of the NS is slowly increased (e.g. 0.3% $\delta\Delta$/million timesteps) until a target Δ=30% is reached. This path through the phase diagram is referred to as Quasiequilbrated Δ Increasing, (QE$\Delta\uparrow$). Then the polydispersity is decreased at the same rate until the system is monodisperse. This path through the phase diagram is referred to as Quasiequilbrated Δ Decreasing, (QE$\Delta\downarrow$). The two paths (QE$\Delta\uparrow$ and QE$\Delta\downarrow$) used are illustrated in Figure 8.6. The simulation is thermostated to a low temperature $T^* = 0.25$ over the entire evolution. Properties, calculated as a function of time, closely approximate those of a CP or AP equilibrated system for a given polydispersity. In regions at T^*=0.25 where the CP and AP phase diagrams differ, the properties calculated by the QE$\Delta\uparrow$ compare favorably to the

AP system and properties calculated by the QE$\Delta\downarrow$ compare favorably to the CP system. If, at a given Δ, the system undergoes a phase transition via nucleation and growth, then this quasi-static method will generally fail to reproduce the equilibrated properties of the system except in the limit where the rate of changing Δ per time step approaches zero. Regions where the quasi-static method fails for this reason will be indicated.

Another reason this method is superior to collecting and averaging a large number of computationally-intensive independent CP simulations at fixed Δ values is the degeneracy found between independent simulations of ordered phases due to finite-size effects. For the H, L, and PLH phase, which do not have triply periodic unit cells, finite simulation sizes allow for the phase to arrange itself inside the simulation box in more than one way (e.g. orientation, distribution of perforations), and at slightly different energies. These differences should disappear at the thermodynamic limit. The self-assembled L phase is also susceptible to bilayer crystal defects, which should also disappear at the thermodynamic limit, but introduce slight energy differences between simulations.

These structural differences between different simulations of the same ordered phase confound the determination of the influence of polydispersity. However, if independent CP polydisperse systems are evolved along a QE$\Delta\downarrow$ path to obtain a desired property value at $\Delta = 0$, the systems all follow identical curves relative to the monodisperse value. The quasi-static method therefore, correctly captures the perturbation polydispersity introduces to properties of the system at its monodisperse thermodynamic limit. The potential energy, packing fraction, and coordination are thus reported as offsets from the $\Delta = 0$ value. This also allows these properties to be more directly compared across volume fractions, ϕ.

8.1.4 Adjusting thermodynamic variables, T^* and Δ

For CP and AP simulations, the thermodynamic variables, T^* and Δ are adjusted as follows.

Temperature is changed by resetting the BD thermostat or (equivalently) by reseting the ϵ pair interaction parameter to change T^* a small amount and then permitting the system to relax to its local equilibrium.

The simulation system sizes studied are large enough that polydispersity can be considered a global system variable. To change the polydispersity, the diameters of every particle in the system is adjusted only a small amount per time step to prevent high energy particle overlaps from occurring. Also, at each time step, the net volume of all the particles is kept fixed so ϕ remains constant as Δ is changed.

To accomplish this, the diameter of each particle i is made a function of λ, $d_i(\lambda)$, where the set of functions $\{d_i\}$ are chosen to be individually monotonically increasing (or decreasing) and to satisfy a constant volume constraint or,

$$\sum_i^N \frac{d_i(\lambda)^3\pi}{6} = V_0. \tag{8.2}$$

The variable λ is increased as a function of time from 0 to 1. The polydispersity thus becomes a function of λ, where

$$\Delta(\lambda) = \frac{100\delta(\{d_i(\lambda)\}}{\hat{\sigma}(\{d_i(\lambda)\})}, \tag{8.3}$$

where δ and $\hat{\sigma}$ are the standard deviation and average of the set of diameters, respectively. In practice, a target distribution $\{d_i(\lambda = 1)\}$ at a given target polydispersity (e.g. 30%) is generated. Then at interim points, $0 < \lambda < 1$, a set of diameters are generated using linear interpolation between initial and final diameter for each particle. The set is then uniformly shifted or scaled at each interim point to satisfy the

volume constraint. Other constraints, such as a minimum NS diameter, can then be imposed, but may necessitate multiple iterations before all constraints are satisfied. Polydispersity can then be changed by adjusting λ by small increments.

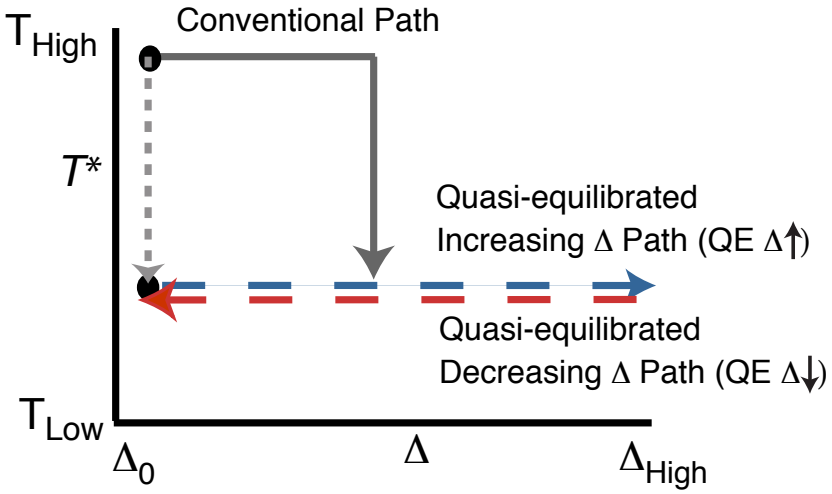


Figure 8.6: Schematic of paths used in determining the quasiequilibration properties of polydispersity

8.2 Analysis techniques

8.2.1 Identification of the double gyroid by the structure factor

To identify the DG structure in our simulations we calculate the structure factor in addition to using visual inspection. The structure factor in a simulation cell with a periodic structure is calculated following[169] for the nanoparticle component of the system:

$$S(\vec{q}) = \frac{\left(\sum_j cos(\vec{q}\cdot\vec{r})\right)^2 + \left(\sum_j sin(\vec{q}\cdot\vec{r})\right)^2}{N} \tag{8.4}$$

where the wave vector, q, is restricted to an integer number of wavelengths within the simulation box, $q = 2\pi\left(\frac{n_x}{L_x}, \frac{n_y}{L_y}, \frac{n_z}{L_z}\right)$. A structure is considered to be the DG if

peaks are identified at $m = \sqrt{6}$ and $\sqrt{8}$[156,157,170] where $m = \sqrt{n_x^2 + n_y^2 + n_z^2}$. Peaks at higher frequencies that are also associated with the DG, namely $\sqrt{20}$ and $\sqrt{22}$ were generally not clearly visible in our systems as they were obscured by noise in the structure factor, however excluding these higher order peaks is reasonable as our calculations are supplemented by visual inspection.

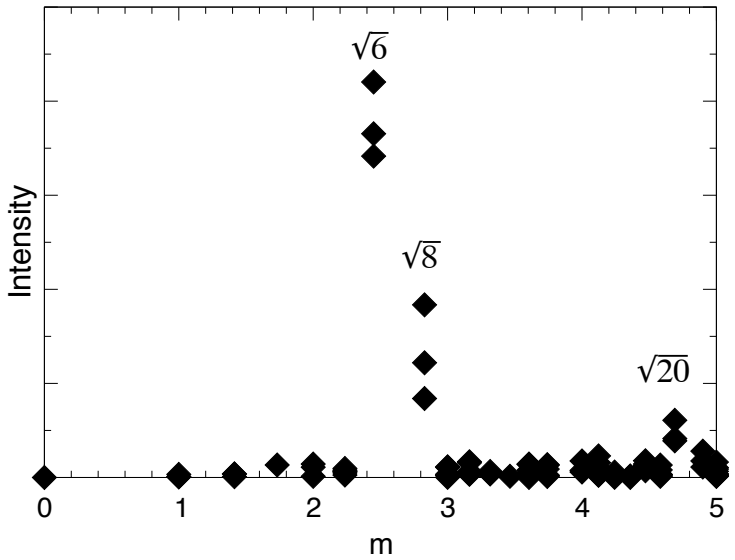


Figure 8.7: The structure factor of a double gyroid with 20% polydispersity is shown as a function of m, the modulus of the integer wavelengths scaled wave vector, which is independent of the unit cell size. The characteristic gyroid peaks at $\sqrt{6}$, $\sqrt{8}$, and $\sqrt{20}$ are clearly visible. This image was originally published in reference[6].

8.2.2 R_{YLM} local structure analysis

To analyze the local configurations of the nanospheres, we performed the R_{YLM} local structure analysis first introduced in Iacovella et al.[25] and further discussed in references[155]. The R_{YLM} method relies on creating a rotationally invariant spherical harmonic fingerprint of the central particle of a cluster of particles (for harmonics l = 4, 6, 12) and then matching this fingerprint to a library of known structures. A

cluster is identified as the reference configuration that minimizes the residual value, R, where $R = \sqrt{\sum_{i=4}^{12}(Q_i - Q_{ref})^2 + \sum_{i=4}^{12}(w_i - w_{ref})^2}$, or it is classified as disordered if it exceeds a certain cutoff. In the definition of R, Q_i, and w_i are two metrics of local ordering based on evaluating a set of spherical harmonic functions $Y_{lm}(\theta, \phi)$ and defined in Steinhardt et al[171]. For our system, the local packing of nanospheres is divided into clusters by grouping each NS and its nearest-neighbor nanospheres. Nearest neighbors are considered to be those with a surface-to-surface distance less than or equal to 0.5 , such that each neighbor would be within the potential well of the central NS. In turn, each NS in the system is considered as a central NS to determine the distribution of cluster types. This method characterizes the bond angles in the local structures rather than the radial distance. The R_{YLM} method permits recognition of the fact that the internal structure of a domain may be composed of many different local structures, not just one dominant structure type. In general, local structures are characterized by the family to which they belong, namely icosahedral, crystalline, or disordered. Each family of structure types contains multiple reference structures with different coordination numbers. To characterize local icosahedral packing, we incorporated into our reference database a series of partial icosahedral clusters that maintain the same bond angles as the full icosahedral cluster, but with 0-4 particles removed. These local structures are almost identical to the LJ minimum potential energy clusters found by Wales and Doye[172], which were also included in the reference database. The local structure is considered to be icosahedral if it matches the partial clusters from Wales and Doye[172] or a partial icosahedral cluster.

The library also includes a family of crystalline structures composed of full and partial coordination clusters with face-centered-cubic and hexagonal-close-packed bond angles.

For the study of binary LJ clusters in section 8.4.2 of this chapter, clusters are also compared to a family of Frank-Kasper (FK) polyhedra with coordinations 8 through

16, referred to as ZN, where N is the coordination number. The Z12 structure is the basis of the icosahedral family. Partial Frank-Kasper polyhedra were not included, as these local configurations, in practice, generally have similar structure and thus similar spherical harmonic fingerprints to partial icosahedra.

The local structure of the DG was analyzed at $T^* = 0.25$ with a residual cutoff of $\sqrt{0.1} \approx 0.316$. Any cluster with an R value greater than 0.316 is considered disordered.

8.2.3 Voronoi tessellation

To examine packing density (compactness) and nearest neighbor trends as a function of polydispersity we use an extension of the Voronoi tessellation. The Voronoi cell around a point is generally defined as the region of space that is closer to the given point than any other point. In a three-dimensional space, the Voronoi tessellation for a set of points uniquely divides the space into irregular polyhedra with flat faces and straight edges. If each point is the center of a sphere in a system of non-overlapping monodisperse spheres, then each sphere will be completely contained within its Voronoi cell. The volume fraction of the sphere inside its Voronoi cell has been proposed as a local measure of density. For polydisperse spheres, a standard Voronoi tessellation is no longer a suitable tessellation since it is possible for the Voronoi cell to be completely embedded inside of a sphere. Instead a radical tessellation, an extension of the Voronoi tessellation, is used to study the packing of the polydisperse head groups. Like the Voronoi tessellation, the radical tessellation also decomposes space into irregular polyhedra with flat faces and straight edges. Each sphere of the polydisperse set will be completely contained within its radical cell. The packing fraction of the head groups is measured by dividing the volume of the NS by the sum of the volumes of the tessellation cells that contain the NS. How this method can be used to study phase separated soft matter systems with complicated geometries is further discussed in Part III, Chapter 9.

We also use the radical tessellation to measure the local coordination or neighbor shell of each NS. We are interested only in the NS neighbors of a NS, and not the tether component, since the NS-NS interaction represents the important energetic interaction in the cooled system. Two NS are considered to be neighbors if the radical cells of the two nanospheres share a facet. We use this definition to calculate both how the system average NS coordination number (CN) is affected by polydispersity.

8.3 Results

8.3.1 Detailed CP phase diagram of Double Gyroid

Cooling tethered nanospheres generated self-assembled DG systems for ϕ from 0.285 to 0.315. To remove any artificial bias from our system, we started from high temperature disordered configurations of polydisperse TNS. Nominal polydispersity values of $\Delta = 0$, 2, 4, 5, 6, 8, 10, 12, 14, 15, 16, 18, 20, 24, 25, and 30% were considered. SA polydisperse DGs formed at values of Δ = 0, 2, 4, 5, 6, 8, 10, 12, 14, 15, 18, 20, 24 and 25% on cooling. For $\Delta > 20\%$, the DG phases generally disassembled within 10 million time steps unless cooled to below $T^* = 0.26$. Systems that did not form the DG were generally found to form hexagonally close packed cylinders (H), perforated lamellae (PL), an intermediate cylinder/perforated lamellae phase (H/PLH), or disordered wormy micelles (DWM). Both H and PL are the neighboring phases for the DG phase in the monodisperse TNS system[25] and they are found at polydispersity levels where the gyroid did not form. The presence of H/PLH and DWM appears to be a TNS system kinetically trapped in a disordered state or oscillating between the H and PL phase. The order-disorder temperature of the SA polydisperse double gyroids for D<10% occurs at $T^* = 0.3 \pm 0.1$. The order-disorder temperature of the SA polydisperse DG for Δ >10% drops to $T^* = 0.26$ for $\Delta \geq$ 25%.

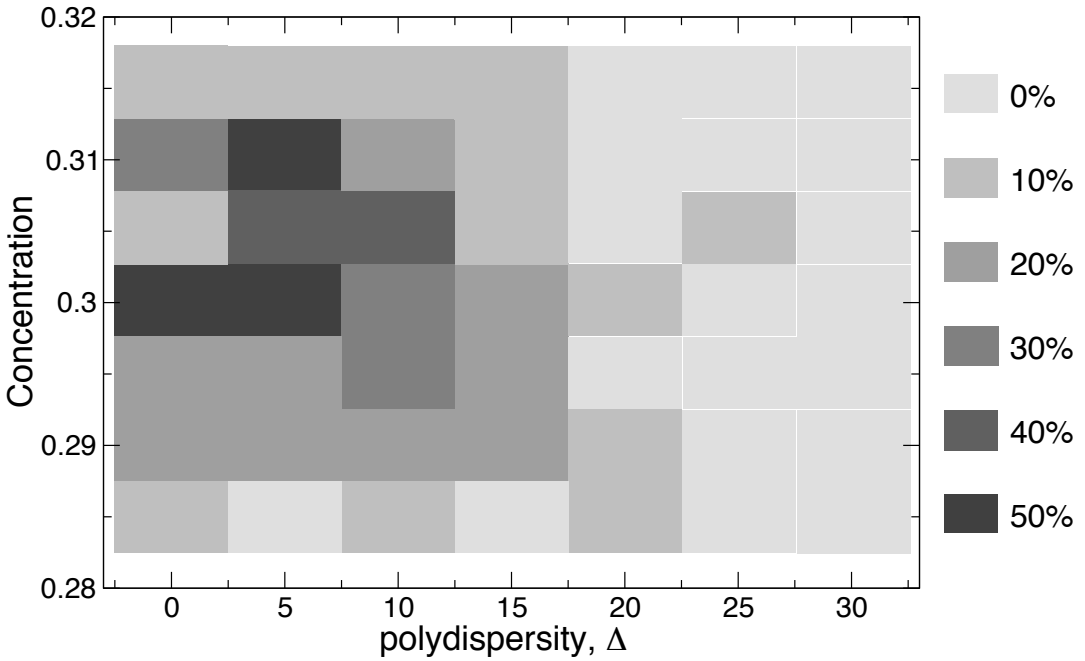


Figure 8.8: A survey of the TNS phase diagram for $0.285 \leq \phi \leq 0.315$ and $0\% \leq \Delta \leq 30\%$, indicating the probability of observing the DG phase. The darkness of the shading indicates the fraction of the ten trials for which the DG phase was found. If multiple simulation box sizes were considered, the box size that produced most instances of the DG phase was used. This image was originally published in reference[6].

Figure 8.8 shows the results of the phase diagram survey for $0.285 \leq \phi \leq 0.315$ and $0\% \leq \Delta \leq 30\%$, plotted in grid fashion with steps 0.005 in volume fraction and 5% in Δ. The relative proportion of the DG phase found at each concentration and (nominal) polydispersity is shown by darkness of shading. To be considered stable, we required that the phase in question persist for a minimum of 10 million time steps.

In this study, a single phase is almost never exclusively found at a state point due to kinetics and metastability. The distribution of alternate phases found (H, PL, and a intermediate H/PLH phase) can be found in the supplemental material. If the assembled structure in a simulation cell contained too many flaws (for example, screw dislocations, non-ordered connections between the cylinders or layers), the simulation

was discarded as being not clearly identifiable. This is consistent with studying the self assembly of small systems very close to the boundaries between phases. In the thermodynamic limit, we would expect only a single phase to be present. We observe that the H phase appears predominantly at lower volume fractions and the PL phase appears predominantly at higher volume fractions in the range examined. As polydispersity is increased, the cross-over volume fraction from H phase to PL phase, as alternate phases, increases.

The DG phase appears to have a stability range of $\phi = 0.3\pm$ 0.01. We observe a peak in the presence of the DG phase for Δ between 5-10%. We also observe that for $\Delta > 15\%$, the DG phase self assembled more rarely, even when using a cooling schedule targeting self-assembly at $T^* = 0.25$. This may indicate that the ideal box size has shifted slightly, that DG with $\Delta > 15\%$ are kinetically difficult to form, or that the DG phase is no longer the free energy minimum phase for the system.

8.3.2 CP and AP phase diagrams

To determine the impact of Δ on the TNS phase diagram, the system was studied at volume fractions that represent each of the self-assembled phases of the monodisperse TNS system studied in Iacovella et al.[25], or the H, DG, PLH and L phases. A volume fraction of $\phi = 0.25$ was chosen to study the H phase. The PLH and L phase, for the monodisperse system, are found at the same volume fraction, but at different temperatures, so the volume fraction $\phi = 0.4$ was chosen to study both of these phases. Small investigations at nearby ϕ confirmed that the behavior of the phase at a single ϕ snapshot is generalizable to other volume fractions where the phase is present. The DG phase is studied at $\phi = 0.3 \pm 0.015$. These results are provided for comparison. To generate the phase diagram, simulations of 505 and 2000 TNS were simulated for $\phi = 0.25$ and 0.4. For the H phase, nominal polydispersities of $\Delta = 0$, 10, 20, and 30% were considered. For the L and PLH phase, nominal polydispersities

of Δ = 0, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 20, and 30% were considered. For each nominal polydispersity a minimum of 5 different random NS diameter distributions were considered. The DG phase was considered at nominal polydisperties of Δ = 0, 2, 4, 5, 6, 8, 10, 12, 14, 15, 16, 18, 20, 24, 25, and 30 % and for $0.285 \leq \phi \leq 0.315$, or 5% above and below where the DG phase was found for the monodisperse case in Iacovella et al.[25] These results are provided for comparison to the rest of the phase diagram.

For the CP and AP phase diagram, a "phase" was considered stable if the phase persisted for a minimum of 10 million time steps. For parts of the phase diagram where the AP and CP phase diagram differed, this criteria was extended to 100 million time steps.

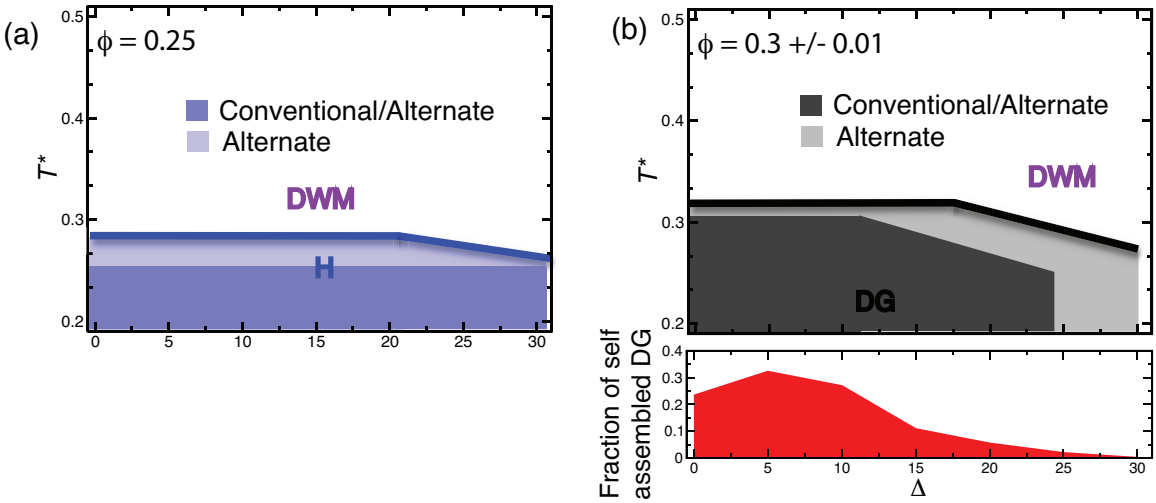


Figure 8.9: The CP and AP phase diagrams (T^* versus Δ) are shown in overlay for the volume fraction (a) $\phi = 0.25$ and (b) $\phi = 0.3 \pm 0.01$. The region where the CP indicates the phases to be stable is contained within the region where the AP indicates the phase to be stable. Thus the darker shaded region is labeled both Conventional and Alternate. The bottom right graph (b) shows for DG the relative likelihood of the DG phase self assembling via the CP as a function of polydispersity.

Using the CP method to generate a phase diagram, a single phase is almost never exclusively found at a state point due to kinetics and metastability. At each volume fraction where an ordered phase is found, some fraction of the simulations become

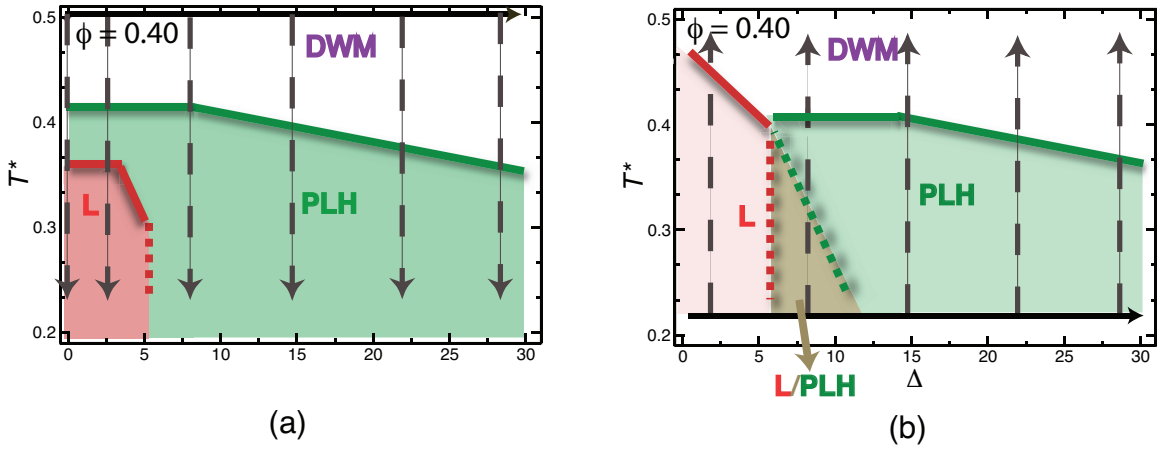


Figure 8.10: The (a) CP and (b) AP phase diagrams (T^* versus Δ) is shown for the volume fraction $\phi = 0.4$. The arrows illustrate the path used to explore the phase diagram.

kinetically trapped in disordered states or ordered phases with defects (e.g. screw dislocations, non-ordered connections between the cylinders or layers). For the DG phase, which exists in a small region of the phase diagram, close to both the H, PLH, and L phase, some fraction of the simulations produce the neighboring phases, or even an intermediate phase with planar connections between the hexagonally packed cylinders. In the thermodynamic limit, we would expect only a single phase to be present. In Figures 8.9 and 8.10, the CP phase diagrams show the most ordered phase found at each state point, where the H and PLH phase are considered more ordered than DWM, L is considered more ordered than PLH, and DG is considered more ordered than H, L, or PLH. The likelihood of forming the DG phase using a CP was found to be a strong function of polydispersity. Therefore, extra consideration is given to the frequency of finding the DG phase relative to polydispersity in determining an "effective" terminal polydispersity.

Using the AP method, a single phase is almost always found at a given state point. The only exception is close to the order-disorder temperature T^*_{ODT} where a small amount of deviation in T^*_{ODT} of individual simulations may occur. This uniformity is

a natural consequence of initializing the simulation in the assumed ordered state for a given T^*, ϕ, and Δ. In figures 8.9 and 8.10, the AP phase diagram show the least ordered state found at each state point.

In Figure 8.9, the CP and AP phase diagrams (T^* versus Δ) are shown in overlay for the volume fraction $\phi = 0.25$ and $\phi = 0.3 \pm 0.01$. For both the H and DG phase, the region where the CP indicates the phases to be stable is contained within the region where the AP indicates the phase to be stable. Thus the darker shaded region is labeled both Conventional and Alternate.

Considering up to $\Delta = 30\%$, the H phase has no terminal polydispersity. However, the AP phase diagram indicates that for $\Delta \geq 20\%$, increasing Δ lowers T^*_{ODT}. From $\Delta = 20\%$ to 30%, T^*_{ODT} decreases from 0.28 to 0.25.

For the DG phase, the CP and AP phase diagram indicates that increasing Δ lowers T^*_{ODT} for Δ >10 and Δ >20% respectively. The CP and AP phase diagram shows the DG phase to be stable up to $\Delta = 25$ and 30%, respectively. However, a study of DG formed by CP indicates that polydispersity can have significant impact on the likelihood of the DG phase forming. In Figure 8.9, the relative likelihood of finding the DG phase at a given polydispersity is determined by dividing the number of DG phase found in simulation at a given polydispersity by the total number of DG phases found in all simulations for $0.29 \leq \phi \leq 0.31$. This data is based on 350 simulations from $\Delta = 0\%$ to 30%, of which 56 formed the DG phase. The data is presented this way to separate the influence of polydispersity on formation of the DG phase from the general kinetic difficulty of forming the DG phase. We observe that a small amount of polydispersity ($\approx 5\%$) helps the DG to form, while for $\Delta > 10\%$, increasing Δ decreases the likelihood of the DG phase forming. For comparison, while fewer total independent simulations were considered for the CP-generated H and lamellar phases, the simulations consistently showed the likelihood of finding the two phases from 0 to 15% was the same as from 15% to 30%.

In Figure 8.10, the CP and AP phase diagram for a TNS system of $\phi = 0.4$ are shown. Arrows indicating the path through the CP and AP phase diagram are provided for illustration.

The CP phase diagram indicates that both the PLH and L phases are present at low Δ, at high and low temperatures respectively, but that increasing Δ lowers T^*_{ODT} and that only PLH is found for Δ >5%. Specifically, as found in reference[24] for the monodisperse system, the CP phase diagram indicates that for $0.375 \leq T^* \leq 0.4$, the PLH phase is present. For $T^* \leq 0.35$, the L phase is present. Over the range $3\% \leq \Delta \leq 5\%$, the T^*_{ODT} for the L phase decreases from 0.35 to 0.3. For $\Delta > 5\%$, the L phase is no longer present, and only the PLH phase is found. For $\Delta > 7\%$, the T^*_{ODT} for the PLH phase also decreases with increasing Δ.

In contrast, for the AP phase diagram, only the L phase is present for low Δ, while for intermediate levels of Δ, a coexistence phase is found at low temperatures and a PLH phase is found at higher temperatures. At high levels of Δ, only a PLH phase is present. Specifically, the AP phase diagram shows the L phase to be present for T^* <0.475 at $\Delta = 0$. For increasing Δ, the T^*_{ODT} for the L phase decreases to 0.4. Between $\Delta = 6\%$ and 12%, at $T^* = 0.25$, the TNS system the L and PLH phase are in coexistence (subsequently referred to as L/PLH), with an increasing fraction of the system in the PLH phase with increasing Δ. For $\Delta \geq 12\%$, only the PLH phase is present. For Δ increasing from 6 to 12%, the T^*_{ODT} of the coexistence region decreases from 0.4 to 0.25. For $\Delta > 15\%$, increasing Δ decreases the T^*_{ODT} of the PLH phase. The cause of the substantial difference between the CP and AP phase diagrams at $\phi = 0.4$ is discussed in Section 5.

A special concern for the AP phase diagram at ϕ=0.4 is that the starting point state for increasing the polydispersity of the NS is a crystal phase. For the H, DG, and PLH phase, the NS are diffusive. For the L phase, where the NS form a bilayer crystal, the NS usually stay fixed in a lattice position over the simulation and the

individual system cannot explore different arrangements of NS. However, we find that by randomly distributing the growth schedules over the NS, the properties of one sampling is not measurably different from another. For example, there is no significant impact on phase boundaries found.

8.3.3 Polydispersity perturbation functions

The stability of a phase is affected by the potential energy of the NS packing, which is, in turn, affected by the average number of NS neighbors, or coordination number, and packing fraction of the phase. Therefore, we are interested in understanding how polydispersity influences these properties. We are also interested in characterizing the composition of local NS ordering, (e.g. crystalline vs icosahedral) as a function of Δ. Also we are interested in characterizing the coexistence region in the AP phase diagram as a function of increasing Δ. To determine how polydispersity perturbs the properties of the monodisperse TNS system, we calculate the properties of a quasi-static system where polydispersity is first slowly increased (QE$\Delta\uparrow$) and then decreased (QE$\Delta\downarrow$).

Figure 8.11 show how polydispersity perturbs the potential energy, packing fraction, average coordination number, crystalline ordering, and icosahedral ordering of the TNS at ϕ =0.25, 0.3, and 0.4, respectively. For potential energy, packing fraction, and coordination number, these properties are shown as the deviation or offset from the monodisperse system value. For coordination number and potential energy, these properties are shown as intensive properties by scaling by the number of NS in the system.

For $\phi = 0.25$, the two quasi-static paths, QE$\Delta\uparrow$ and QE$\Delta\downarrow$, have a minimal amount of hysteresis relative to each other, so the two paths have been combined for each property. Also, the system is in the H phase over the entire polydispersity range. The same minimal hysteresis and presence of a single (DG) phase holds true

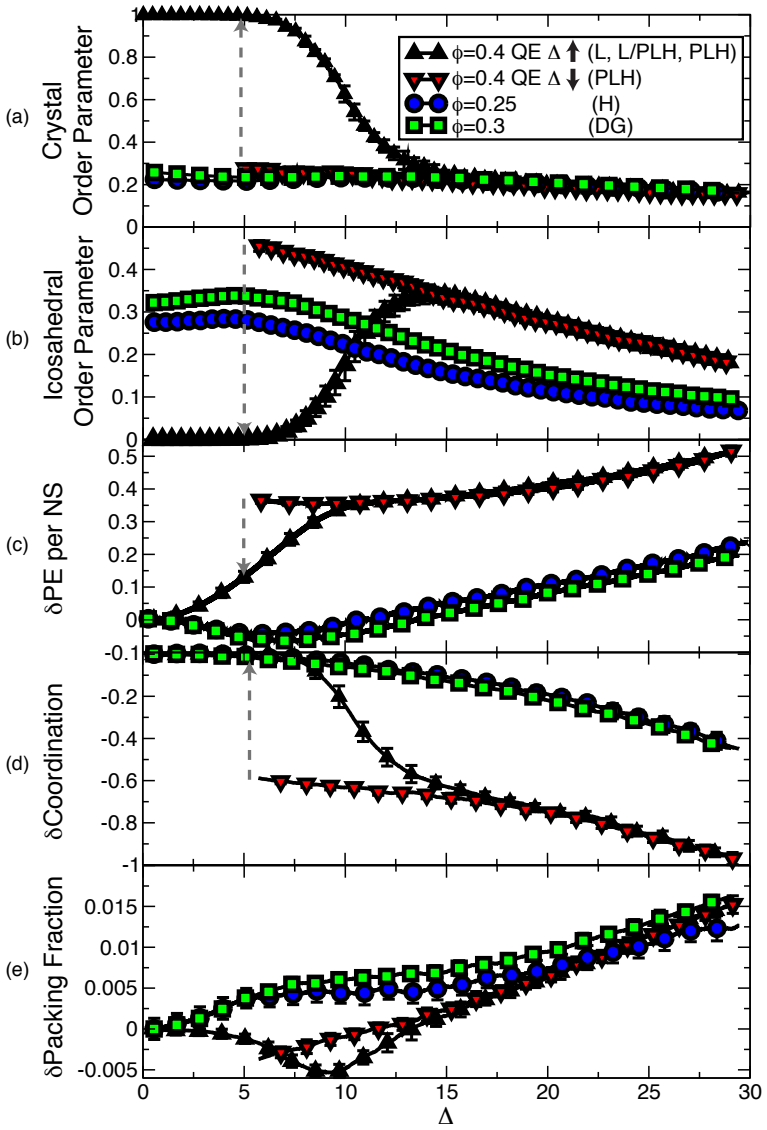


Figure 8.11: The effect of polydispersity on (a) the crystalline vs (b) icosahedral local packing, (c) the potential energy of the NS-NS interaction, and (d) the coordination and (e) packing fraction of the particles, as determined by a radical tessellation, is compared at ϕ=0.25, 0.3, and 0.4. The phases found along each curve are indicated in parentheses in the legend. The dashed arrows indicates the collapse of QE$\Delta\downarrow$ data onto QE$\Delta\uparrow$ data at $\phi = 0.4$.

at ϕ=0.3, and so the two quasi-static paths have also been combined .

For $\phi = 0.4$, the L, PLH, or L/PLH phase can each be present as both a function of Δ and path. Subsequently, at $\phi = 0.4$, the two quasi-static paths, QE$\Delta\uparrow$ and QE$\Delta\downarrow$,

have a minimal amount of hysteresis for $\Delta > 12\%$ (where both CP and AP phase diagrams indicate the PLH phase is present), but a significant amount of hysteresis for $\Delta < 12\%$. For $\Delta < 12\%$, properties calculated from QE$\Delta\uparrow$ compare favorably to the AP phase diagram and properties calculated from QE$\Delta\downarrow$ compare favorably to the CP phase diagram. However, for the QE$\Delta\downarrow$ path, for Δ <5%, crystallization L phase occurs so slowly relative to the rate of decreasing the polydispersity, that the quasi-static method proves poor for generating statistics. Therefore, for QE$\Delta\downarrow$ $\Delta <$ 5%, data is not shown. A grey dashed line is used to indicate that the properties of the system transition to the QE$\Delta\uparrow$ curve, which was verified at a few points by changing the rate of decreasing polydispersity to zero and permitting the system to equilibrate. For each figure, properties of the $\phi = 0.25$ (H phase) and $\phi = 0.3$ (DG phase) are shown as blue circles and green squares respectively. The $\phi = 0.4$ QE$\Delta\uparrow$ data is shown as black triangles pointed up. The $\phi = 0.4$ QE$\Delta\downarrow$ data is shown as red triangles pointed down. For the $\phi = 0.4$ QE$\Delta\uparrow$ data, by $\Delta = 12\%$, the system is in the PLH phase. All of the $\phi = 0.4$ QE$\Delta\downarrow$ data shown is in the PLH phase.

In Ref.[25], Iacovella et al. divided the TNS phase diagram into two general regions, one characterized by liquid-like icosahedral NS packing, (i.e. DWM, H, DG, and PLH phase), with the fraction of icosahedral local ordering increasing with increasing ϕ, and a second characterized by crystalline NS packing region (L phase). In Figures 8.11(a) and (b) the impact of Δ on the crystalline and icosahedral local ordering is considered, respectively. In fig. 8.11(a), a crystalline order parameter of $\approx$0.2 indicates a non-crystalline system for this parameterization of the R_{YLM} analysis (e.g. residual cutoff, library of structures, temperature of system). As expected, the $\phi = 0.25$ and $\phi = 0.3$ systems (H and DG phase) have no crystalline structure. The local crystalline ordering of these two systems is measured at $\approx$0.2 for all Δ. For, QE$\Delta\uparrow$ the $\phi = 0.4$ system has crystalline structure until 6%. The fraction of the system in local crystalline ordering then decreases until by $\Delta \approx 12\%$, no local

crystalline structure is measured for increasing Δ. For QE$\Delta\downarrow$, no crystalline structure is measured for 5%< Δ <30%.

In Figure 8.11(b), local icosahedral ordering is shown to be promoted slightly at $\Delta = 5\%$ for the H and DG phase. Increasing Δ greater than 5% disrupts the local icosahedral motif. The $\phi = 0.4$ QE$\Delta\uparrow$, shows no icosahedral local ordering until 6%. Consistent with the decreased presence of the crystalline phase in Fig.8.11(a), icosahedral ordering increases until $\Delta \approx 12\%$ and then decreases with increasing Δ. For 5%< Δ <30%, increasing Δ decreases the icosahedral local ordering for $\phi = 0.4$ QE$\Delta\downarrow$. Figures 8.11(a) and (b) clearly indicate the coexistence of L and PLH for $\phi = 0.4$ QE$\Delta\uparrow$, as well as the lack of a coexistence region found for $\phi = 0.4$ QE$\Delta\downarrow$ data.

In Figure 8.12, snapshots of the system at $\phi = 0.4$ are shown in the coexistence region. NS that are identified as having a crystalline, icosahedral, and unidentified bond ordering by the R_{YLM} measure are colored red, dark blue, and light blue respectively. As Δ increases from 0 to 12%, we observe the icosahedral/liquid regions emerging initially along grain boundaries and defects. At $\Delta = 10\%$, the crystal region has become an "island" in disordered PLH regions.

In Figures 8.11(c)-(e), we see that the impact of Δ on system properties is primarily determined by the composition of the local packing structure, icosahedral or crystalline, as measured by Figures 8.11(a) and(b). The properties of phases with a liquid-like, icosahedral local ordering behave similarly upon an increase in Δ and distinctly differently from the L phase with local crystalline ordering. Properties of the coexistence phase behave as an average of the properties of icosahedral and crystalline phases, weighted by their proportional presence, with increasing Δ.

In Figure 8.11(c), we consider the contribution to the potential energy of the system from the NS-NS interactions for the various polydisperse phases. This energetic interaction, which neglects the NS-tether and tether-tether energetic interactions, will

be subsequently referred to as the potential energy (PE) of the system. In Figure 8.11(c), the average offset to the PE per NS is shown as a function of Δ. For ϕ =0.25 and 0.3 (the H and DG phase), the PE of the NS first decreases with increasing Δ and then, between $\Delta = 6 - 8\%$, the potential energy increases with increasing Δ. In contrast, for the $\phi = 0.4$ QE$\Delta\uparrow$ data, PE increases with increasing Δ. There is no observable trend change in PE at the transition from the L to the L/PLH coexistence phase. However, between $\Delta = 8\%$ and 12%, the PE increases more slowly. For $\Delta > 12\%$, when the system is in the PLH phase, the PE increases with increasing Δ at the same slope as the H and DG phase at $\phi = 0.25$ and 0.3. The QE$\Delta\downarrow$ (PLH phase), although offset, shows the same trend relative to increasing Δ as the H and DG phase.

In Figure 8.11(d), the deviation to average NS coordination number from the monodisperse value is shown for $\phi = 0.25$, 0.3, and 0.4. For the H and DG phase, the average coordination number smoothly decreases with increasing Δ. For the $\phi = 0.4$ QE$\Delta\uparrow$, increasing Δ has a small impact for 0< Δ <6%. For 6< Δ <12%, the average coordination number decreases strongly, and then continues to decreases a less steep slope for 12 <Δ< 30%. For the $\phi = 0.4$ QE$\Delta\downarrow$, the deviation to average coordination number decreases with increasing Δ. Although offset, the trend of the $\phi = 0.4$ QE$\Delta\uparrow$ data, for 15 <Δ< 30%, and the $\phi = 0.4$ QE$\Delta\downarrow$ data, for 5< Δ <30% is similar to the H and DG phase data. We conclude that impact of polydispersity on the average coordination number (relative to the monodisperse value) is invariant to phase, but that the average coordination of the icosahedral liquid-like PLH phase at $\phi = 0.4$ is $\approx$0.6 less than the crystaline bilayer phase at the same Δ.

In Figure 8.11(e), the deviation to packing fraction of the NS domain relative to the monodisperse value is shown for ϕ=0.25, 0.3, and 0.4. For $\phi = 0.25$ and 0.3 (the H and DG phase), the NS domain becomes denser with increasing Δ. For the $\phi = 0.4$ QE$\Delta\uparrow$ data, increasing Δ causes the NS domain in the L phase to become less dense,

or swell, between 0%< Δ <6%. For 6%< Δ <12%, in the coexistence phase, the differences between swelling crystalline and densifying icosahedral liquid components results in a net packing fraction mimimum to occur at $\approx$ 9% and then increase for 9%< Δ <30%. For the $\phi = 0.4$ QE$\Delta\downarrow$, the packing fraction increases with increasing Δ. Although offset, the trend of the $\phi = 0.4$ QE$\Delta\uparrow$ data, for 15< Δ <30%, and $\phi = 0.4$ QE$\Delta\downarrow$ data, for 5< Δ <30% is similar to the H and DG phase. No gray drop-line is provided at $\Delta = 5\%$ for the $\phi = 0.4$ QE$\Delta\downarrow$ data, as it is apparent that the onset of crystallization is coincident with the two phases (L and PLH) being at the same approximately density.

8.3.3.1 Discussion of coexistence between L and PLH phase

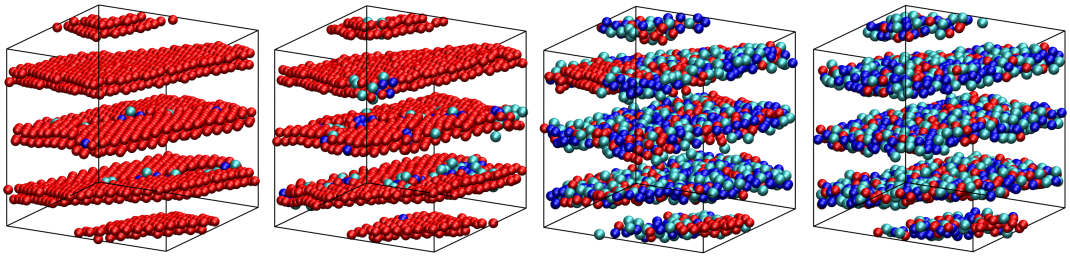


Figure 8.12: A series of snapshots of the L phase is shown at polydispersity 0%, 6%, 10%, and 12% as polydispersity is grown into a monodisperse cooled system. At each polydispersity, the system was allowed to relax for 10 million time steps. Nanospheres are colored red if locally crystalline, light blue if unidentifiable, and dark blue if icosahedral. Tethers are not shown. Initially the system is in a totally crystalline state, with a few non-crystal NS found at grain boundaries and defects in the lamallae. At 6%, the system is still crystal, although the number non-crystal NS at grain boundaries and lamellae defects has increased. At 10%, the lamallae is mostly liquid and disordered, with a few small islands of crystal bilayer remaining. At 12%, the system is fully in the PLH phase. Note that although "red" particles are still present, they are not spatially correlated and represents the limitations of the identification algorithm.

In comparing the AP and CP phase diagrams at ϕ=0.25, 0.3 ± 0.01, and 0.4, the H, DG, and PLH (for $\Delta > 12\%$) phases show only slight path dependence. Each phase is present up to a higher T^* in the AP phase diagram, and the DG phase is

present at a higher Δ. The crystalline bilayer, L phase, in comparison, has a clear path dependence. The L phase is present over a larger area of the AP phase diagram and a L/PLH coexistence region is present on the AP phase diagram, but not the CP phase diagram. To understand why the coexistence region is present in the AP phase diagram but not the CP phase diagram we consider the literature on the phases found in polydisperse sphere systems.

Most studies of polydispersity have focused on hard disc and hard sphere systems[162,166,173–176] with a few exceptions[177]. As the repulsive $\frac{1}{r^6}$ term of the LJ potential acts similar to a hard sphere, polydisperse hard-sphere literature can be appealed to as a first order approximation of a system of Lennard-Jones spheres confined to lamellar domains.

The maximum polydispersity of the solid crystalline phase in systems of 2D hard discs has been shown to be around 8%[178] with a possible coexistence region between 8% and 15%[173]. In comparison, a study of a two dimensional lattice of LJ particles found a terminal polydispersity at 6% for a low density crystalline solid that increased to 9.7% as density was increased. For a 3D system of hard spheres, Kofke and Bolhius[162,174] found the solid phase has a terminal polydispersity of $\Delta = 5.7\%$, but could coexist with a liquid phase of up to 11.8% polydispersity at increased pressure. For hard spheres, Kofke and Bolhius found stable coexistence regions of liquid and crystal, Auer and Frenkel[175] found that, above 5%, polydispersity increasingly inhibits the formation of critical crystalline nuclei. In experiments, the formation of crystallites in a polydisperse (>5%) system has been observed albeit in a sheared suspension and when permitted to equilibrate over hours to days[179,180].

The bilayer crystalline system has a combination of the characteristics of a two-dimensional and a three-dimensional system of hard spheres. It is strongly constrained in two directions parallel to the lamellar plane and softly constrained in the direction perpendicular to the lamellar plane. However, the bilayer lamella is also capable of

reorienting inside the simulation cell (within limits) to relieve stress in the planar directions.

Using a Conventional Path, the crystalline bilayer did not self-assemble for $\Delta >$ 5% polydispersity. For $\Delta > 3\%$, the amount of simulation time required to self-assemble the crystalline bilayer noticeably increased. Using the Alternate Path, the crystalline bilayer was stable up to 6%, and, at low temperature a stable crystal/liquid phase was present until 12%. We hypothesize that the coexistence phase found by the AP represents the free energy minimizing state, but that the polydisperse NS kinetically inhibit the phase from forming within the times accessible to our simulations. In an experimental realization of this system with access to longer equilibration times, the coexistence of liquid and crystalline lamellar phases may be found.

While the coexistence region of the AP phase diagram are found over the same ranges of polydispersity predicted by Kofke and Bolhius for hard spheres, we recognize that the composition of the phase may be different from a spontaneously self-assembled state. Kofke and Bolhius predicted fractionation in the liquid/crystal system. That is, the solid phase was predicted to be composed of slighly larger spheres. Other researchers[166,176] have also predicted that a polydisperse fluid could fractionate and crystallize into two crystals, each with sufficiently low polydispersity to nucleate. Growing polydispersity in a frozen system excludes size fractionation. Also, unlike the liquid phases (H, PLH, DG) which have measurable diffusion coefficients, all relaxation is local and there is rearrangement of the NS beyond original lattice sites. However, multiple runs with different random starting distributions all showed a nearly identical response to a quasi-equilibrated increase in Δ. And in the portion of the CP and AP phase diagrams where the same phase in predicted (e.g. $\Delta < 5\%$ and $\Delta > 12\%$ Δ), all measurements were identical.

8.4 Local Structure Analysis of the Double Gyroid

We can better understand the trends in polydispersity by analyzing the effect of polydispersity on the packing properties of the NS. We use the R_{YLM} analysis to identify the local structure, and calculate the average packing properties of the DG domain using a Voronoi (radical) tessellation.

8.4.1 The R_{YLM} local structure analysis

It was observed that in monodisperse TNS icosahedral structures are favored in systems of NS confined to cylindrical geometries where the relative diameter of the cylinder is less than 5, otherwise hcp/fcc crystalline arrangements form22. As previously noted, the DG phase is essentially composed of a series of interconnected cylindrical tubes. In the TNS system, the tether sterically restricts particle packing and the NS tend to pack into icosahedral and crystalline clusters with partial coordination (i.e. one coordination position of an ideal cluster must be unoccupied so that the central particles tether can escape the local structure). The R_{YLM} method is used to match the pattern of bond angles between a particle and its nearest neighbors, or coordination shell, with a library of structural motifs. Iacovella et al.[25] showed using the R_{YLM} method with a residual cutoff R = 0.316 that at $T^* = 0.256$, about 30% of the NS are central particles of a local structure that resembles an icosahedral cluster with partial coordination. Since Iacovella et al. also proposed that the local icosahedral packing stabilized the DG structure, it is of interest to consider how adding polydispersity to the system affects the local packing.

In Fig. 8.11a and 8.11b, the local structure analysis of polydisperse gyroids is shown at T^*= 0.25. We observe a distinct peak in icosahedral ordering at $\Delta = 68\%$ and almost no icosahedral ordering at $\Delta = 20\%$. For $\Delta > 20\%$, the icosahedral packing motif is disrupted by polydispersity.

In general we note that polydispersity increases the dimensionality of the energy

landscape for the NS. Individual NS are no longer interchangeable. However, for low levels of polydispersity most of the NS are still the same size and are approximately interchangeable, i.e. the energy difference caused by interchanging the particles should be small. We find the formation of icosahedral local packing is still dominant at low polydispersity. In fact, a low level of polydispersity promotes well-ordered icosahedral local structure, with the degree of icosahedrality peaking at approximately $\Delta = 68\%$. At higher polydispersity, the formation of local icosahedral packing is rapidly suppressed. We note that this trend is consistent with the idea that local icosahedral packing stabilizes the DG structure as local icosahedral packing becomes suppressed at roughly the same polydispersity level where the DG ceases to self-assemble readily from a disordered configuration (i.e. $\Delta > 15\%$).

8.4.2 Analysis of low polydispersity promotion of local icosahedral packing.

	Binary Ratio					
Coordination	1.05	1.10	1.15	1.20	1.25	1.30
8	I 0.091	I 0.103	I 0.097	I 0.130	I 0.146	I 0.149
9	I 0.056	I 0.017	I 0.070	I 0.147	Z 0.153	Z 0.146
10	I 0.040	I 0.014	I 0.196	I 0.208	I 0.261	Z 0.049
11	I 0.005	I 0.004	I 0.062	I 0.108	Z 0.147	Z 0.234
12*	I 0.001	I 0.105	I 0.115	I 0.217	I 0.198	I 0.219

Figure 8.13: A cluster analysis is performed on the energy minimizing binary clusters of Doye 2005[7] and the Cambridge Cluster Database. For each binary cluster, an I is indicated if the cluster best matched a full or partial icosahedral cluster, or a Z for best matching a Frank-Kasper polyhedra. The c value of the cluster match is also shown. For a coordination of 12, note that an icosahedral cluster and the Frank-Kasper polyhedra are identical. This image was originally published in reference[6].

In Figures 8.11b we observe that a low polydispersity promotes the icosahedral packing motif. Locally, in a fluid, a low level of polydispersity implies most particles

are the same size, with an occasional larger or smaller particle present. As such, we consider the impact of the presence of a single larger or smaller particle on the isolated icosahedral cluster.

Doye and Meyer studied the energy minimizing arrangements of isolated binary clusters, i.e. clusters of LJ particles of two different sizes[7]. They found that up to a diameter ratio of 1.1, the low-energy binary cluster formed by 9 - 13 LJ particles is a partial to full icosahedral cluster with all large particles in the shell and the small particle at the center. This arrangement is because the distance between atomic centers for neighboring atoms in the shell of an icosahedral monodisperse cluster is 5.15% larger than that for a central atom and a nearest neighbor atom. A 9.79% reduction in the diameter of the central atom relieves this strain,[7,181] and an energy minimum of the binary icosahedral cluster occurs around this value. However, a further increase in the ratio of the diameters of the two species causes a change in the lowest-energy LJ cluster structure,[7] i.e. small particles begin appearing in the shell as well*, or the partial icosahedral structure becomes distorted. In Table 1 (Figure 8.13) , we perform a R_{YLM} analysis on the lowest-energy binary LJ clusters. We find the clusters progressively deviate from icosahedral bond angles and begin resembling other Frank-Kasper polyhedra as the binary ratio increases.

Following the work of Doye and Meyer, we would expect that for an icosahedral cluster with a single large particle, the large particle would tend to occupy the shell and not the center of the cluster; we would also expect the resulting cluster to be lower energy than an equivalent monodisperse cluster as the larger particle would help relieve strain.

We test this hypothesis by growing a set of NS with discrete binary polydispersity into a monodisperse DG structure. One binary DG was created with 10% of the NS

*The full icosahedral cluster (coordination 12) has a shell of large particles around a small particle up to a binary diameter ratio of 1.15. At a ratio of 1.2, four of the shell particles become small, as the central particle is now too small for twelve large particles to fit around comfortably.

having radii 10% larger and a second was created with 10% of the NS having radii 10% smaller. These binary DGs were created identically to the artificially grown DGs described in Section III at $T^* = 0.256$, and equilibrated at this temperature until the NS had, on average, diffused at least halfway across the simulation box. We find that small NS are 16 times more likely to be found at the center of local icosahedral structures than the large NS. The large NS were 1.3 times more likely to be found in a coordination shell than a small NS. Small NS and large NS promote local icosahedral structure by 2% and 15%, respectively. Small NS may also promote local icosahedral structures with slightly higher coordination; quenched local icosahedral structures had 4% higher coordination. In general, we find that the presence of small NS lowers the potential energy of a given local icosahedral structure by creating a lower-energy, higher-coordination structure, while large NS, which relieve the strain in the coordination shell, encourage more local icosahedral structures to form. Thus, for a DG formed at low polydispersity levels, the minor fraction of smaller and larger NS are working in concert to lower the energy and further stabilize the DG structure. Fig. 8.11d shows that the fraction of well-ordered icosahedral clusters peaks at around a polydispersity of 6%. At this level of polydispersity, we find that a NS in the center of the local icosahedral structure is 5.6% smaller than its coordination shell NS.

8.4.3 Studying the average structure properties with the Voronoi tessellation

We use a generalized Voronoi tessellation, specifically the radical tessellation[32], to determine how polydispersity affects the average volume fraction of the gyroid domain and the average NS coordination number (see section IIIC and the supplemental material for more details). As shown in Fig. 8.11e, we find that as the polydispersity of the system is increased, the gyroid domain becomes more densely packed, increasing by 0.015 from Δ=0% to $\Delta = 24\%$. This increase is a result of the extra degree of

freedom in particle size, which permits the NS to locally arrange in tighter configurations. This is in good agreement with simulations of hard spheres, where variation in particle size is shown to increase the packing fraction[182]. Like the measure of potential energy, calculating the packing fraction of the DG is indifferent to how the polydisperse gyroid is formed. We note that while in monodisperse systems tighter sphere packing is associated with lower energy configurations, the ability of a polydisperse system to pack tighter than a monodisperse system does not necessarily imply that the polydisperse system must also have a lower energy than the monodisperse system. We find that packing fraction increases over the entire range of polydispersity (Fig. 8.11e) while potential energy initially decreases with increasing Δ up to Δ=8% and then increases with increasing Δ (Fig. 8.11c).

Fig. 8.11d shows that increasing polydispersity lowers the average NS coordination number of the system. Linear fits of the NS coordination as a function of the polydispersity below and above 8% are shown to illustrate the slope change that occurs with increasing polydispersity. The potential energy of the NS is a function of two properties, the coordination number of NS and the distance to each neighbor in the coordination shell. For $\Delta < 8\%$ we previously observed that PE decreases with increasing Δ (Fig. 8.11c); we additionally observe that the packing fraction of the DG increases rapidly but the average NS coordination number drops only slightly with increasing Δ. We conclude the system is lowering its PE by finding tighter configurations where more NS are sitting in the bottom of the potential energy wells of their neighbors, in a manner analogous to the isolated icosahedral cluster with a 9.7% smaller particle at its center. As the average NS coordination begins to drop more rapidly, the effect of this drop can be seen in the increase of potential energy shown in Fig. 8.11c.

Although the average NS coordination decreases with increasing polydispersity, this decrease is not universal for all NS sizes, but represents a net effect. The fact

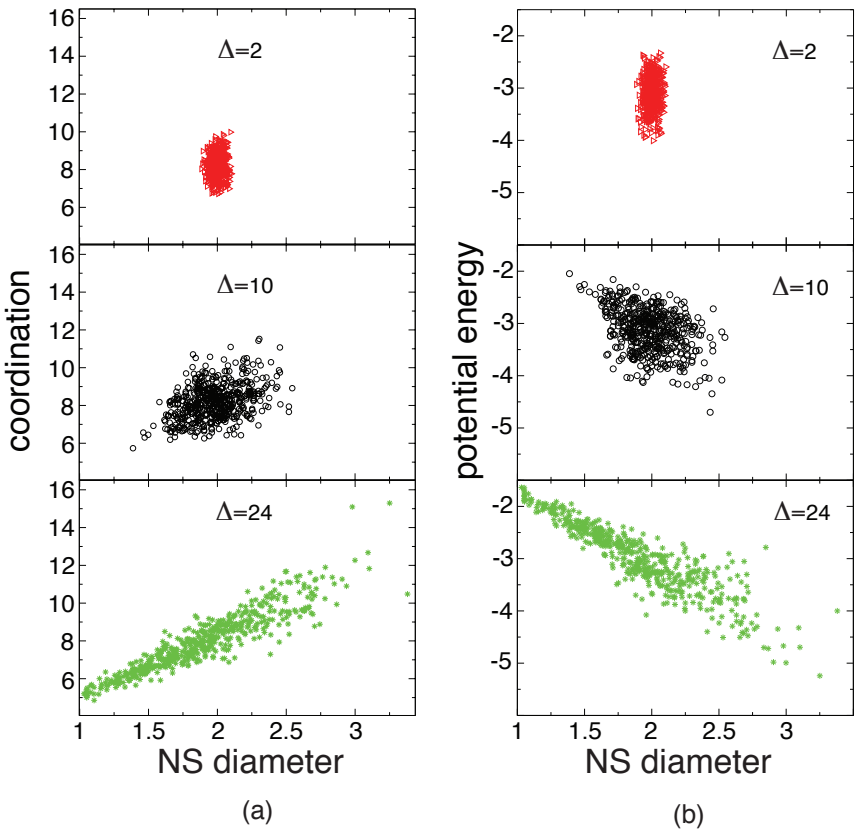


Figure 8.14: Increasing polydispersity induces a spreading in the coordination number and potential energy of the NS in the DG as a function of NS diameter. In (a), the number of NS neighbors (averaged over 2×10^6 time steps) for each NS is shown. In (b), the potential energy (averaged over 2×10^6 time steps) for each NS is shown. This image was originally published in reference[6].

that the net effect is negative may be due to sphere packing in a DG structure. In Fig. 8.14a, we show the coordination number for the NS in different polydisperse systems averaged over 20,000 time units and plotted against the diameter of each particle. As the polydispersity increases, the coordination number becomes a strong function of NS diameter. In Fig. 8.15, we show how average coordination trends with polydispersity and how the coordination becomes a stronger function of sphere size as polydispersity increases. The correlation coefficient, which measures the strength of the linear relationship between diameter and number of neighbors, increases significantly between $\Delta = 2$, 10, and 24%. As the potential energy of a shifted Lennard-

Polydispersity	Average NS coordination number	Correlation coefficient
2%	8.24 ± 2.00	0.080
10%	8.23 ± 2.02	0.395
24%	7.98 ± 2.25	0.922

Figure 8.15: An analysis is performed on the influence of polydispersity on the average coordination and the correlation between NS diameter and coordination for the data shown in Fig. 8.14.This image was originally published in reference[6].

Jones NS is a strong function of its coordination, in Fig. 8.14b we see that the average PE of each NS, averaged over the same elapsed time, decreases with NS diameter. It is also clear that for the more polydisperse DG, the vertical spread of potential energy values for a given diameter is a function of the particle diameter. This reflects the different diffusion coefficients for large and small particles in the same system, as shown in Fig. 8.16. That is, smaller NS, in shallower potential energy wells, diffuse faster through the system and also explore the range of possible energy configurations faster, resulting in less spread in the measured potential energy for a small diameter.

We conclude that for low levels of polydispersity, the system is able to relieve internal packing frustration, i.e. NS are able to adjust so that more spheres are sitting in the bottom of the potential energy wells of other particles, lowering the energy of the system. For larger values of polydispersity, the net decrease in average NS coordination is responsible for the net increase in the potential energy of the system.

8.5 Conclusion

For the DG, we find that a small amount of polydispersity ($\Delta = 5\%$) encourages the formation of the DG phase, but a large amount ($\Delta > 10\%$) may kinetically prevent the phase from forming. In comparison, for the H and PLH phase, we conclude that

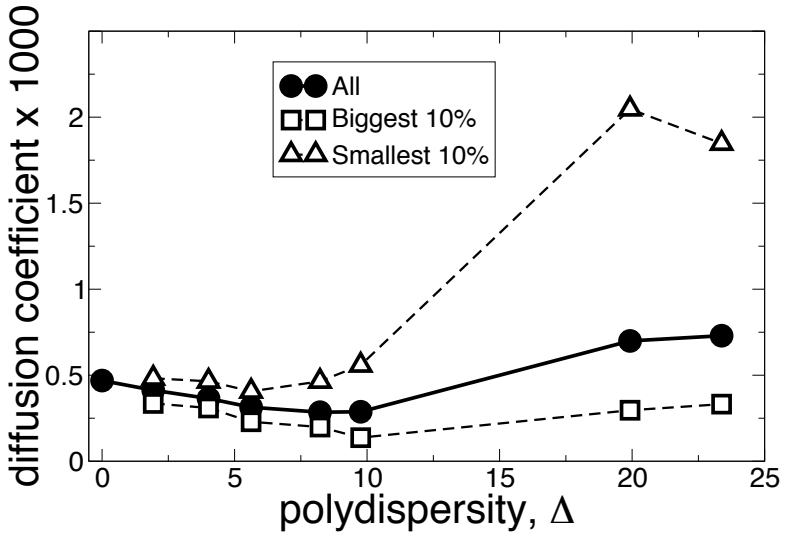


Figure 8.16: The diffusion coefficients of the average, 10% smallest, and 10% largest NS of the DG are shown as a function of polydispersity. The y-axis is scaled by a factor of 1000. The dimensionless units are in $\sqrt{\epsilon/m}$. This image was originally published in reference[6].

there is no terminal polydispersity. Considering as high as Δ=30%, these two phases were found to be present on both the CP and AP phase diagram. The AP phase diagram, however, suggests that above a threshold polydispersity for each phase, the T^*_{ODT} may decrease with increasing Δ. For the PLH phase, a minimal amount of polydispersity ($\Delta = 6 - 12\%$) may be necessary for the phase to be present. For the L phase, we find that the crystalline bilayer phase has a terminal polydispersity of $\approx$ 6%. For 6%$< \Delta <$12%, an L/PLH coexistence region or the PLH phase may be present.

One consequence of the minimal and terminal polydispersity ranges identified above is that a manufactured population of TNS at a given polydispersity value may not be able to form all the ordered phases.

The use of the Alternate Path, AP, phase diagram proves a useful way to study perturbations to a phase diagram. In this case, the perturbation was NS polydisper-

sity. Rather than independently self-assembling perturbed systems, self-assembled ordered systems can be slowly adjusted to the perturbed state. In the case of polydispersity, this easily allows the impact of the perturbation to be studied continuously. Compared to CP phase diagrams, or phase diagrams generated by self-assembly from a disordered system, we found the AP method to reproduce the same phase boundaries with some small deviations. Even features particular to the CP phase diagram, such as the larger PLH region, can be reproduced by the AP method by considering an AP phase diagram where the polydispersity is increased to a terminal value, and then decreased again. This method is most valuable for studying phases that are difficult to self-assemble for kinetic reasons, such as the DG phase or the L/PLH coexistence region.

More generally, it is apparent that polydispersity can have subtle but important impacts on the properties of sphere packing, especially in unusual domain geometries. A relatively small amount of polydispersity can disrupt internal structures and change the per-particle energy of the packing. The local packing character of the nanoparticles was the dominant consideration in determining how polydispersity influenced the properties of the phase. In general, the phases with the more disordered packing were more tolerant of polydispersity than the more ordered phases. For sensitive phases that occupy a narrow volume fraction range of the phase diagram, such as the DG phase, or phases that occupy a wide volume fraction range but with internal crystalline ordering, such as the L phase, it can be critical to consider how much polydispersity a phase can tolerate before assuming the phase will be found in experimental systems.

CHAPTER 9

Voronoi Tessellation for Characterizing Microphase Separated Soft Matter Systems

This chapter corresponds to publication:
Phillips, Glotzer, Voronoi Tessellation for Characterizing Microphase Separated Soft Matter Systems, Preprint

Voronoi cells[33] have a rich history of use in the characterization and analysis of soft matter systems[183–190]. They have been used to characterize liquid and solid phase structure by considering the statistics of the Voronoi network[183] and by considering the distribution and shape of the Voronoi polyhedra[184,186] and the distribution of the neighbor statistics[185]. Voronoi cells have been used to measure liquid and vapor densities and identify interfaces in systems with coexistence[187–189]. They have also been used to study structure and dynamics in glass-forming liquids[190].

In this chapter, we propose how Voronoi tessellations can be used to study systems with microphase separation. In microphase separated systems, an ordered mesophase emerges when amphiphiles, or chemical compounds created from immiscible components, minimize their free energy. Classic examples of microphase separation are found in block copolymer systems[191–193]. However recent studies have considered how shape amphiphiles, created by functionalizing anisotropic nanoparticles, may assemble into more complex structures with liquid crystal-like internal ordering[5,29,122,148,155,194]. When studying these microphase separated systems, it is useful

to be able to understand how different state points and variation in particle parameters affect properties such as the density of the different domains or the internal coordination of particles in a domain. Individual amphiphiles in these systems are usually modeled as a collection of spheres connected by a combination of rigid and soft bonds. Some amphiphiles are also modeled by spheres of two or more different sizes. We introduce how a Voronoi tessellation can be used to study such systems. In Section 9.1, we review the Voronoi tessellation and consider applications of the tessellation to an example systems, a double gyroid phase self-assembled from tethered nanorods[155]. In Section 9.2, we discuss a practical method for extending the Voronoi tessellation to systems with a distribution of sphere sizes using existing open source software. In Section 9.3, we will consider a case study of a double gyroid phase formed from particles with a distribution of sphere sizes from Part III, Chapter 8. We will demonstrate how the extended Voronoi tessellation measure provides a useful tool for understanding how the system density responds to temperature and for comparing this DG phase to other self-assembled DG phases. In Section 9.4, we make concluding remarks regarding the use of this measure.

9.1 Voronoi Tessellation

The Voronoi cell around a point is generally defined as the region of space that is closer to the given point than any other point in the system. In a three-dimensional space, the Voronoi tessellation for a set of points uniquely divides the space into irregular polyhedra with flat faces and straight edges. For a system of monodisperse spheres, a set of points can be defined by the center of each sphere, and the resulting Voronoi tessellation can be used to study the packing properties of the system. If the monodisperse spheres are non-overlapping then each sphere will be completely contained within its Voronoi cell and the volume fraction of a sphere inside its cell has been proposed as local measure of density[37,187,195]. In a system of spheres packed

in a face-centered-cubic or hexagonally-close-packed arrangement, for example, the volume fraction of an individual sphere in its Voronoi cell reflects the bulk volume fraction of the packing. Also, counting the faces of a spheres Voronoi cell provides a measure of the local coordination, or neighbors, of an individual sphere in a packing.

Voronoi cells are additive. If a structure, for example, is composed of multiple spheres, then the region of space that is closest to the structure in the space is simply the union of the Voronoi cells formed around each sphere that composes the structure. This is true, even if the multiple spheres are overlapping. We exploit this principle to define the Voronoi volume of a microphase-separated systems domain. In Section 9.2, we will exploit this principle again to approximate a polydisperse, or diameter distribution, generalization of the Voronoi tessellation, the Voronoi S tessellation.

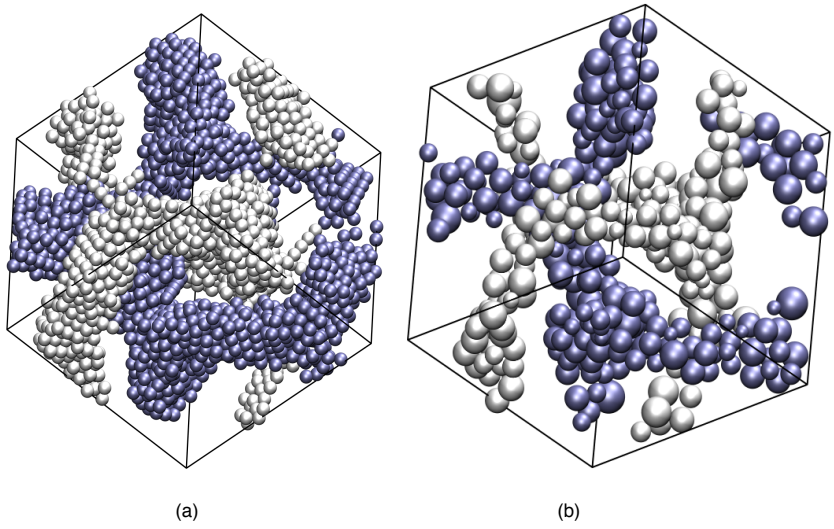


Figure 9.1: (a) A double gyroid phase self-assembled from tethered nanorods. (b) A double gyroid phase self-assembled from 18% polydisperse tethered nanopheres

In a microphase separated system, where different components of the system have separated into different domains, the net Voronoi tessellation of each domain can be used as a measure of the volume of each domain. The net Voronoi volume of a domain is defined as the sum of the Voronoi volumes of each component member.

These Voronoi cells are usually connected to each other, forming the domain, although this is not a necessary requirement. The convenience of this measure is that it is indifferent to the topology of a particular domain, it is rigorously defined, and it can be used to measure how changing parameters such as temperature or particle size impacts the compactness or packing fraction of different domains in the system. Also, the external facets of a tessellation around a domain can be thought of as approximating the intermaterial dividing surface (IMDS) between the phase separated domains. By considering the volume of the entire domain, the packing fraction of just the component in the domain can be characterized. By considering individual Voronoi cells, the local packing fraction of a part of the domain can be characterized at the most meaningful degree of resolution.

In a microphase separated system where one interaction is attractive and all other interactions are essentially volume excluding, it is expected that the aggregating component will represent the densest part of the domain and that as temperature is lowered or epsilon is increased, that the volume occupied by the aggregating component will decrease. Equivalently, as temperature is lowered, it is expected that the packing fraction of the aggregating component domain will increase. Often the packing fraction trend is difficult to quantify in microphase separated systems by creating a simple envelope of a domain. For example, in the case of the double gyroid, a triply periodic, tricontinuous phase, trying to separately characterize the packing fraction of the node versus a channel[25,156] is onerous. The common method of measuring local density by calculating the particle intersections within a cutting sphere envelope, which involves arbitrary choices of sphere size and placement, work best when the cutting sphere envelope is significantly larger than the particles, but smaller than the structure the particles compose. A domain in a microphase separated system may only be a few particles in cross section, causing this method to be inexact. The Voronoi tessellation is indifferent to these issues.

We generated standard three-dimensional Voronoi cells by using the open-source software vor++*, which uses a plane cutting algorithm[37,195] to rapidly and robustly calculate Voronoi cells for three-dimensional systems.

In the left panel of Figure 9.1, we show an example of a microphase separated soft matter system[155], a double gyroid phase that has self-assembled from a system of rods (5 beads, $\sigma = 1.0$) functionalized by a tether at one end (2 beads, $\sigma = 1.0$). More details on this phase can be found in Iacovella, 2008[155]. By considering the union of the Voronoi cells comprising a single domain, it is possible to simply measure the volume fraction of a domain relative to the whole system. Previously the IMDS of DG phases have been modeled using level surfaces or constant mean curvature (CMC) surfaces[196]. The volume fraction of a minority component domain in experiments has been measured by comparing TEM images to level surface or CMC surface models[157]. Using a Voronoi tessellation, and studying a single snapshot of the tethered rod system, we find that the volume fraction of a single minority component of the tethered rod DG phase is 21.8%. The volume fraction measured is higher than the experimentally observed range for block copolymers (13-19%)[196]. The packing fraction in the rod domain, defined as particle volume divided by rod domain volume is 0.343, which is roughly three times as dense as the packing in the tether domain, (packing fraction of 0.107).

In Figure 9.2, we consider the distribution of packing fractions and coordinations for individual rods in a single snapshot of the system. In this case, the packing fraction of each rod is measured as the volume of the spheres composing each rod, $\frac{5\sigma^3\pi}{6}$, divided by the sum volumes of the Voronoi cells for the the spheres. The rod coordination is measured as the number of neighboring rods that create the facets of the union of the set of Voronoi cells for the spheres. In Figure 9.2(a), we observe that the nanorods are not tightly packed, significantly below the fcc/hcp packing fraction

*http://math.lbl.gov/voro++/

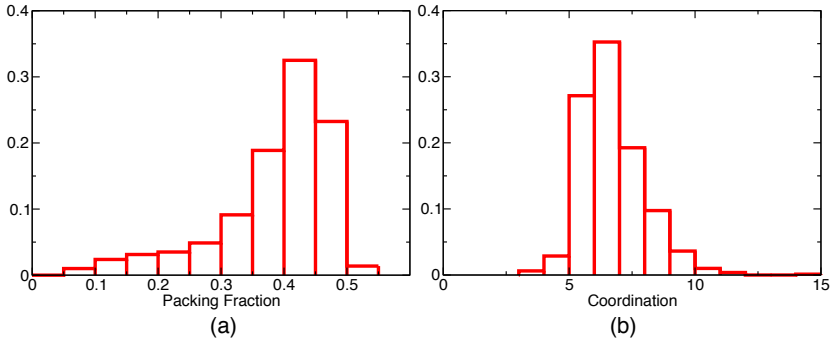


Figure 9.2: (a) The distribution of packing fractions for the rods. (b) The rod-rod coordination for the nanorods

of 0.7405, and, at the densest, approaches the packing fraction of a simple cubic system at 0.52. The outlier, lowest packing fraction nanorods, are nanorods that failed to join the channels and nodes of the DG phase, and are floating in the tether region. Figure 9.2(b), we find that there is a significant distribution of coordinations, but that the distribution is strongly peaked around 6 and 7 rods, as would be expected from a hexagonal rod packing. We find that the average coordination of a rod in the nanosphere domain is 7.215.

In Figure 9.3, a density map cross-section of a unit cell of the tethered rod DG is shown based on the standard Voronoi tessellation. For the nanorods and the tethers, the local densities are shown per-sphere rather than per-rod or per-tether. The high-density regions correspond to the aggregating nanorod component. The highest packing fraction an individual Voronoi cell can achieve in a monodisperse sphere system is the Voronoi cell of a sphere at the center of a local fully-coordinated icosahedral structure[197], or approximately 0.7547. The nanorod domain which is densest at the center of the domain, has peak packing fractions at 0.572. We observe that the relatively larger nanosphere domain compared to other DG phases may be explained by the range of local densities found within the nanorod domain, which is tightly packed towards its center but loosely packed towards the surface of the domain, due

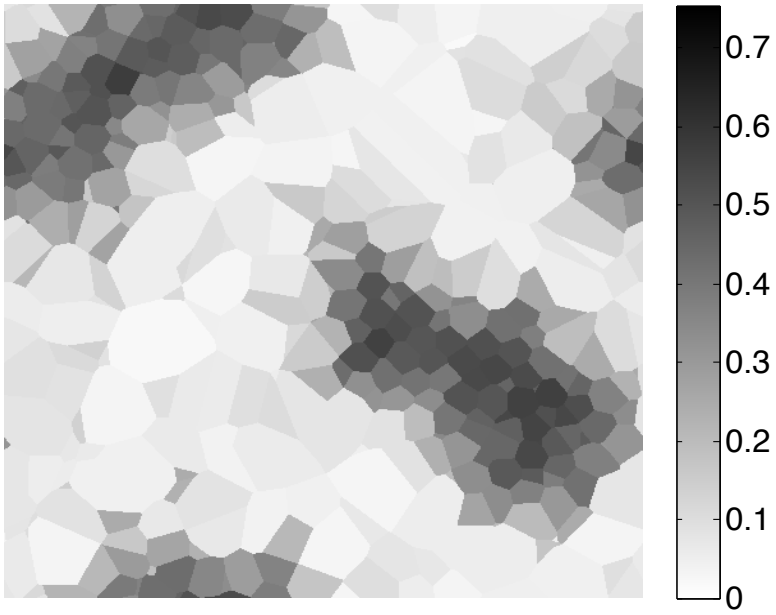


Figure 9.3: A cross section of the density map of a unit cell of a tethered rod double gyroid based on the standard Voronoi tessellation.

to the twisting and skewing of the rods. The tethers, on the other hand, are loosely packed and a considerable amount of void space is present in the tether domain.

Another self-assembled double gyroid phase, shown in the right panel of Figure 9.1, was introduced in[25], and further studied in[155]. This DG phase, which self-assembles from polymer tethered nanospheres where the nanospheres are aggregating, is modeled by a minimum of two different sphere sizes. In order to study this DG phase shown in the right panel of Figure 9.1 the standard Voronoi tessellation must be extended. In the following sections we will show how known extensions of the standard Voronoi tessellation can be practically calculated, and use these tessellations to study the tethered nanosphere double gyroid phase.

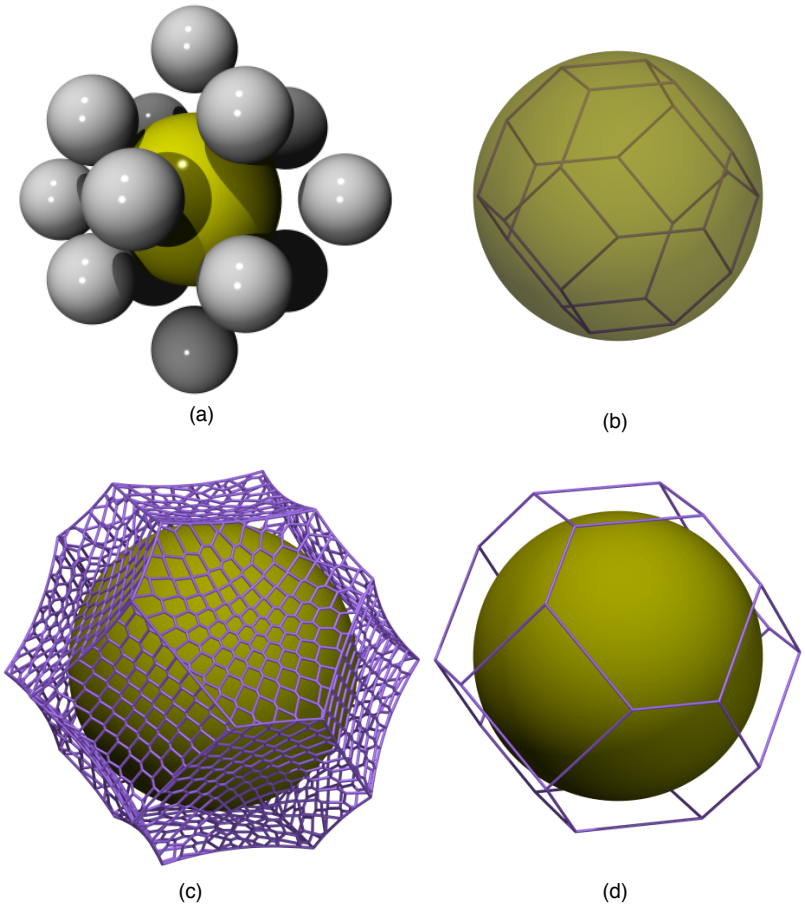


Figure 9.4: (a) We consider a sphere of diameter 2.0 surrounded by smaller spheres of diameter 1.0. (b) A conventionally defined Voronoi cell is embedded inside the large sphere, which has been made partially transparent so the Voronoi cell is apparent. (c) Alternately a Voronoi S cell can be constructed around the sphere, which is shown with a mesh to make the curvature of each face more apparent. (d) Or a radical cell can be constructed around the sphere.

9.2 Voronoi S Cell and Radical Tessellation

In a system of polydisperse spheres, the standard Voronoi tessellation is not a physically appropriate tessellation. Consider, for example, a large sphere surrounded by smaller spheres, as shown in Figure 9.4a. The faces of the Voronoi cell, calculated as bisecting planes between the centers of each pair of spheres, will cut through the larger sphere's surface (fig. 9.4b). In a system of non-overlapping polydisperse

spheres, it is possible, therefore, to obtain a local volume fraction greater than one. Two generalizations of the Voronoi tessellation that have been proposed for systems with polydisperse spheres include the Voronoi S cell, or additively-weighted Voronoi diagram, and the radical tessellation[198] which are illustrated in Figure 9.4c and 9.4d, respectively. The Voronoi S cell of a sphere includes all points that are closer to the surface of the sphere than any other sphere surface. The radical tessellation[32] includes all points whose tangential distance to the surface of the sphere is smaller than the tangential distance to any other sphere surface. While the Voronoi S cell[32,199] has a more physically useful definition, the surfaces of the Voronoi S cell are curved hyperboloids and therefore, volumes must be approximated. The radical tessellation, on the other hand, decomposes space uniquely into flat-faced polyhedra and is therefore simply calculated and its volume can be exactly determined. In a system of monodisperse spheres, both the Voronoi S tessellation and radical tessellation reduce to the standard Voronoi tessellation.

To generate a radical tessellation we made minor modifications to the plane cutting algorithm developed by Rycroft. The plane cutting algorithm calculates the Voronoi cell around a central point by first, initializing the cell as the size of the entire space, and then considering points (cutting points) in successive concentric shells around the initial point and making plane cuts to the cell. When the next concentric shell to be considered is farther from the central point than twice the maximum distance, $r_{Voronoi\_max}$, from the central point to a vertex of the current Voronoi cell, the bisecting plane of any subsequently considered point cannot cut the Voronoi cell, and the algorithm is terminated. Note, subsequent versions of the vor++ improved this algorithm with an extra directional test, that can be ignored for the purpose of this discussion. We extend this algorithm to calculate a radical tessellation by projecting each cutting point radially from the central point by a factor γ times the distance between the cutting and central points. The resultant Voronoi cell cut is equivalent

to the radical cell. The factor γ is defined as follows:

$$\gamma = \left(1 + \frac{r_c^2 - r_i^2}{R^2}\right) \tag{9.1}$$

where r_c is the radius associated with the central point, r_i is the radius associated with the cutting point being projected, and R is the nominal distance between the two points. The criteria for terminating the algorithm must be adjusted, since it is possible that a more distant cutting point could be projected inwardly. In our modification, the algorithm now terminates when rshell is greater than twice $r_{Voronoi_max}$, where r_{shell} is defined as

$$r'_{shell}\left(1 + \frac{r_c^2 - r_{max}^2}{r_{shell}^2}\right) \tag{9.2}$$

Here r_{max} is the maximum radius of any sphere in the system, and r_{shell} is the distance to the next concentric shell.

The ability to calculate radical tessellations has been incorporated into the vor++ open-source code.

To approximate the Voronoi S cell of spheres in a polydisperse system, each sphere is decomposed into roughly 800-1500 overlapping smaller spheres, embedded inside the larger sphere and touching the surface of the larger sphere at points distributed evenly over the surface of the larger sphere, using the golden section spiral method to distribute the points[200],[†]. A single small sphere is placed at the center, to prevent the creation of a single highly degenerate vertex. The size of the small spheres used for the decomposition is chosen to be the size of the smallest sphere in the system. This decomposed system now consists of monodisperse spheres (albeit, overlapping) and a standard Voronoi tessellation can be applied. Here we employ the principle that the Voronoi cell for an object that can be decomposed into elements whose Voronoi cells can be calculated is equal to the union of the Voronoi cells of the individual elements.

[†]For an explanation as to why to use a golden section based generalized spiral, see http://www.ogre.nu/pack/pack.htm

The Voronoi S cell for the original large sphere is approximated as the union of all the standard Voronoi cells of the small spheres into which it was decomposed. As the number of smaller spheres in the decomposition is increased, their union converges to a Voronoi S cell. For each face of the Voronoi S cell dividing the space between the central spheres and a neighboring sphere, it theoretically would be sufficient to decompose the larger of the two into a collection of spheres the size of the smaller of the two. This, however, adds complexity to the algorithm.

The convergence of this approximation to a Voronoi S cell was verified by comparing the location of the vertices generated with those generated by a corrected implementation of an algorithm by Medvedev[‡ 199]. In a non-degenerate system of polydisperse spheres, three and only three edges and faces meet at vertex. Each vertex of the Voronoi S network is equivalent to the center of a constructed interstitial sphere that is simultaneously tangent to four spheres. Medvedevs algorithm constructs an initial vertex and then follows the curved edges of the network, assuming it is simply connected, until all vertices have been found. Medvedev's algorithm does not generate individual Voronoi cells from which a volume could be simply calculated.

The Medvedev algorithm was applied to a unit cell, of length 23.8031 on each side, of a double gyroid phase consisting of large spheres of $\sigma = 2.0$ and a small spheres of diameter $\sigma = 1.0$. The ratio of large to small spheres was 1:8. To approximate the Voronoi S cell tessellation, each of the large spheres was decomposed into 720 overlapping spheres of $\sigma = 1.0$. Medvedevs algorithm[199] identified 30,083 unique vertices in the binary sphere system. Of the vertices generated by the decomposition methods, 99.88% were generated by both the same quadruple of spheres and were within a distance of 3e-3 of the vertex generated by Medvedevs algorithm[199].

In general, the calculation of a Voronoi S cell tessellation required the calculation

[‡]Correspondence with Dr. Medvedev clarified that the vector l, whose dot product with vector t_{ijk} is used to determine the correct direction to proceed along the Voronoi S-Channel, should be drawn from the tangent point of the interstitial sphere centered on the site with the first ball i, to the tangent point of this sphere with the fourth ball l . A misprint in the text omitted this explanation.

of approximately two orders of magnitude more standard Voronoi cells as compared to the radical tessellation. Also, the termination criterion for the Rycroft plane-cutting algorithm described above is optimal for nearly-spherical Voronoi cells. When a large sphere is decomposed into numerous small spheres, the shape of the standard Voronoi cell around each small sphere is extremely oblong in shape in the large sphere radial direction. This increases the number of iterations of the plane-cutting algorithm before termination.

In the next section, we will show that the radical tessellation and the Voronoi S cell tessellation behaved similarly with regard to the properties being studied (Figure 9.7 and Figure 9.8) with, generally, a fixed offset between the two measures.

9.3 Characterizing the Microphase Separated Domain of a Double Gyroid with the Voronoi Tessellation

Iacovella et al.,[25], showed that a tethered nanosphere, modeled as a large sphere of $\sigma = 2.0$ (the nanosphere) attached to a string of eight small spheres of diameter $\sigma = 1.0$ (the tether beads), self assembles into a double gyroid (DG) phase at a system concentration of 0.3 when the larger spheres are aggregating. Further details of the model assumptions can be found in Iacovella et al.[25]. A question of interest is how nanosphere polydispersity effects this phase. Note that a "monodisperse" TNS system is a binary, not monodisperse, system of spheres, as the tether beads and nanospheres are different sizes. We will now contrast the radical tessellation to the Voronoi S tessellation for studying such a system.

In Figure 9.5 and 9.6, a density map cross-section of a unit cell of a monodisperse and polydisperse DG, $\Delta = 24\%$, is shown based on both the radical tessellation and the Voronoi S tessellation. The high-density regions correspond to the aggregating nanosphere component. The highest packing fraction an individual Voronoi

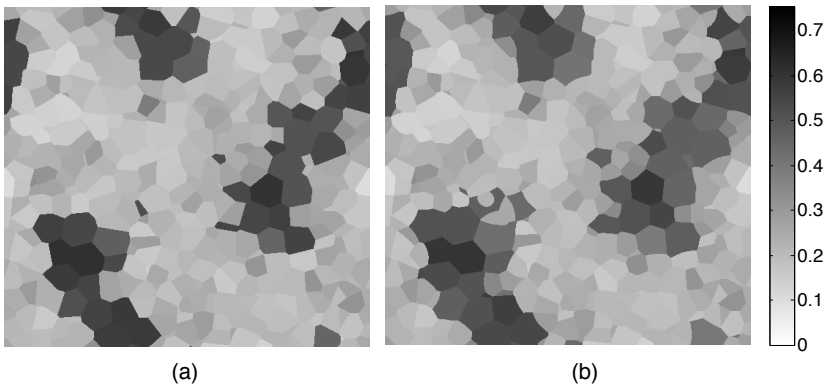


Figure 9.5: (a) A cross section of the density map of a unit cell of a monodisperse gyroid based on the radical tessellation and (b) based on a Voronoi S tessellation. Shading indicates the volume fraction of the Voronoi cell cut in the cross section.

cell can achieve in a monodisperse system is the Voronoi cell of a sphere at the center of a local fully-coordinated icosahedral structure, or approximately 0.7547[197]. In the monodisperse density cross-section, cross-sections of a local partial icosahedral structure can be observed. In a polydisperse system, the local packing fraction in a volume cell can theoretically approach 1.0 as polydispersity increases, though these configurations are unlikely to form in a DG. In the monodisperse DG system, where generally only partially coordinated local icosahedral structures are possible due to the steric constraints, the peak packing fraction (i.e. local density) is approximately 0.650, if measured by a Voronoi S tessellation, or 0.659, if measured by a radical tessellation. In block copolymer systems, the formation of voids in the node is believed to destabilize a double gyroid phase. The node of a block copolymer double gyroid is composed of many aggregating monodisperse spheres (beads) tethered to each other. The concentration profile of the node of a block copolymer system in the double gyroid phase[156], can locally approach the fully coordinated icosahedral structure packing fraction, but not across the entire node. The DG node in a TNS system is composed of fewer, larger spheres with different steric constraints than a node composed of chains of six monodisperse polymer beads from the aggregating

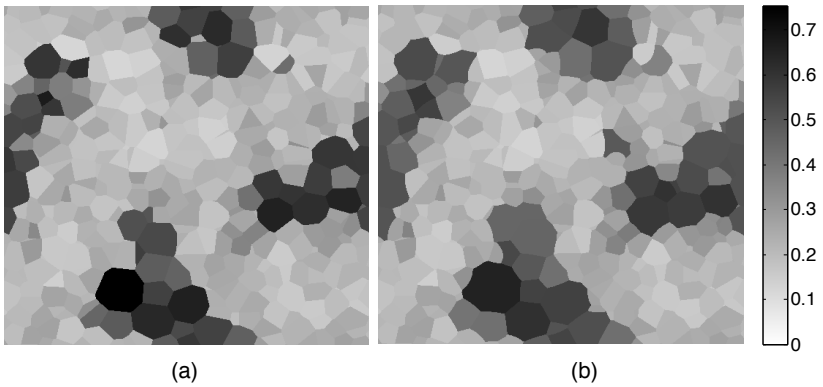


Figure 9.6: (a) A cross section of the density map of a unit cell of a double gyroid with polydispersity 24% based on the radical tessellation, (b) and based on a Voronoi S tessellation. Shading indicates the volume fraction of the Voronoi cell cut in the cross section.

portion of a diblock copolymer. Namely, the DG node in a TNS system is not large enough, relative to the particles comprising it, to form voids in the node.

In Figure 9.7, the impact of temperature on the volume of the entire DG domain in a monodisperse DG is considered. As expected, this measure shows that the DG domain becomes more compact as the temperature of the system is lowered; that is, the volume occupied by the aggregating component decreases. Conversely, the packing fraction of the aggregating component within its domain increases with decreasing temperature. A Voronoi S tessellation and a radical tessellation demonstrate similar trends with regard to packing fraction, just offset by a fixed difference. From Figures 9.5 and 9.6, it is apparent that the Voronoi S tessellation generally assigns more volume to the large sphere at a large-NS/small-tether-sphere interface. Therefore, since in this system the aggregating NSs are always larger than the tether spheres, the Voronoi S tessellation will always calculate a lower packing fraction, with respect to the DG domain, than the radical tessellation. The two tessellations show similar trends, with a fixed difference between their calculations.

As in Section 9.1, we use the union of the Voronoi cells comprising a single domain to simply measure the volume fraction of a domain relative to the whole system. The

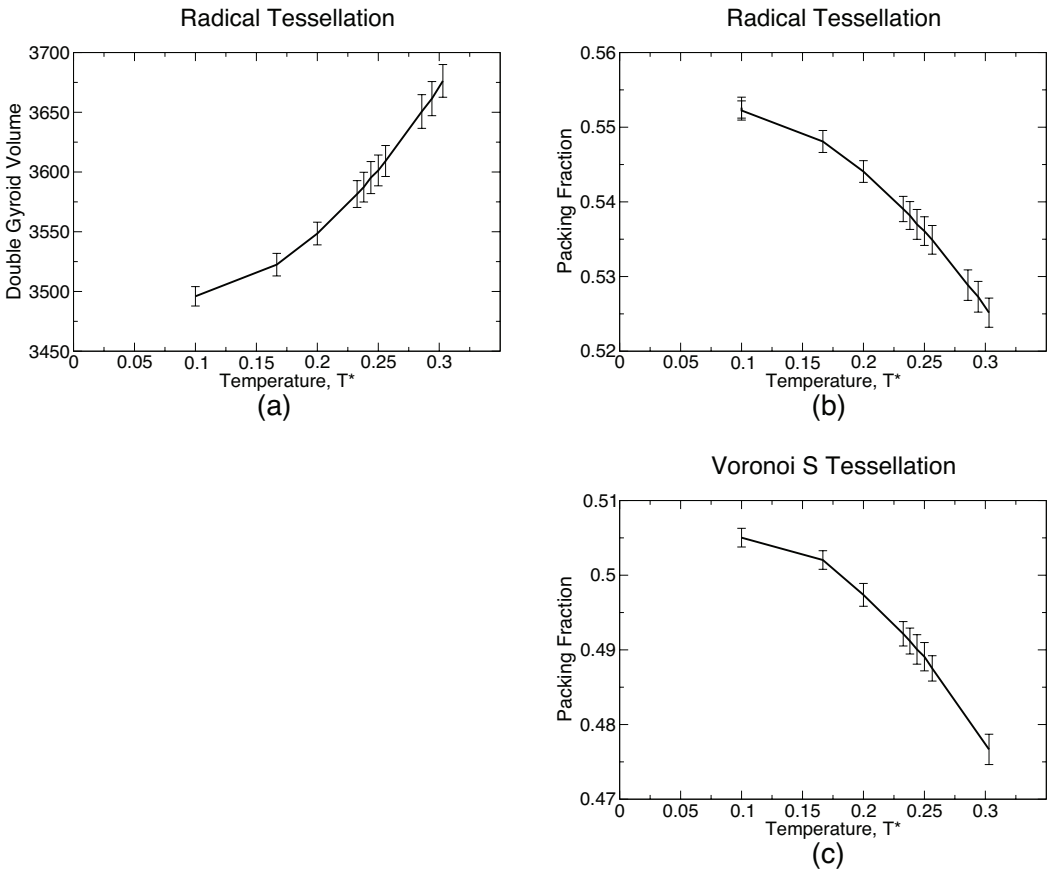


Figure 9.7: (a) The radical volume of a monodisperse gyroid domain in a unit cell as a function of temperature. (b) The packing fraction of the monodisperse gyroid domain as a function of temperature, as calculated by dividing the volume of the nanoparticles by the radical volume of the gyroid. As expected, lowering the temperature causes the NS to pack tighter. (c) The packing fraction of the monodisperse gyroid domain based on a Voronoi S tessellation.

radical tessellation measures the volume fraction of one of the two minority components as 13-14%, varying as a function of temperature. The Voronoi S tessellation, which assigns slightly more volume to the large sphere at a large NS sphere/small tether sphere interface, measures the same minority component volume fraction as 14-15%. The volume fraction measured is near the lower bound of the experimentally observed range for block copolymers (13-19%)[196].

The measure of the average packing fraction of a domain can be sensitive to even small changes to the system. For example, a monodisperse gyroid with nanospheres

1.07% smaller in diameter than the standard diameter of 2.0, was grown and equilibrated in a simulation cell with an unchanged box length, effectively lowering the volume of the net nanospheres by 3.2% and the concentration of the system to 0.295. As the DG is periodic in all three dimensions, the DG cannot reorient within its simulation box to relieve the stress, so the DG has to stretch. This DG was measured to have domain density of 0.531 +/- 0.002, or 0.8% less dense than the DG with the standard sized nanospheres.

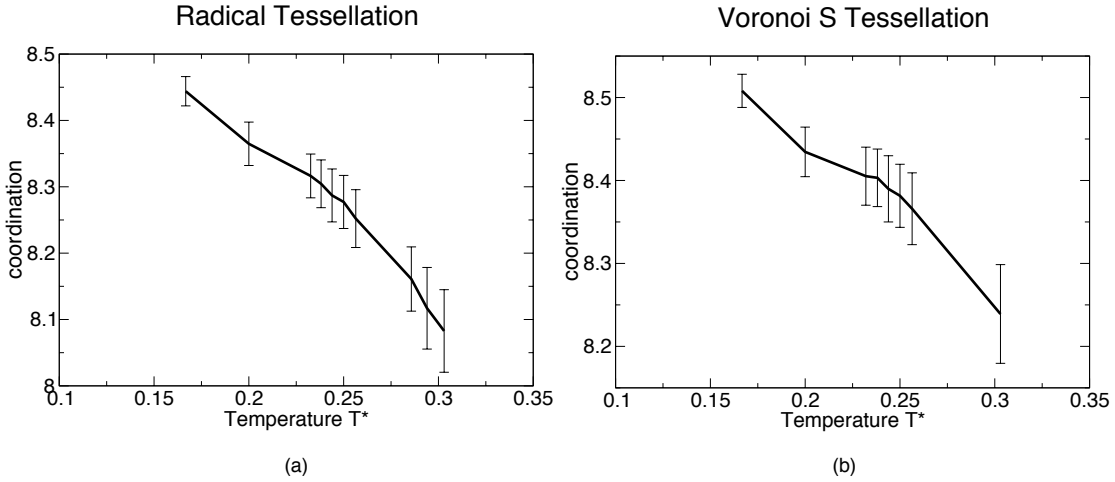


Figure 9.8: (a) The average NS coordination based on a radical tessellation of monodisperse DG as function of increasing temperature. Increasing temperature causes a slight downward trend in the average number of neighbors. (b) The average NS coordination based on a Voronoi S tessellation.

In Figure 9.8 we show the average neighbors, or coordination, of a nanosphere in a monodisperse gyroid as a function of temperature. Neighboring nanospheres are identified by the internal facets of a union of Voronoi cells comprising a domain, and is a measure that is specific to the tessellation used. As the temperature of the system decreases, the average nanosphere coordination slightly increases. Again, the radical tessellation and the Voronoi S tessellation have similar trends, but with a fixed difference between them. The Voronoi S tessellation assigns approximately 0.1 more neighbors per nanosphere than the radical tessellation, consistent with also

assigning a slightly larger cell. We have observed that the two tessellations showed similar trends, with a fixed difference between their calculations, as a function of polydispersity as well.

In Part III, Chapter 8, we studied a polydisperse tethered nanosphere system. The radical tessellation is used to study the influence of polydispersity on the packing fraction of the NS domain.

9.4 Conclusion

We conclude that a Voronoi tessellation is a useful tool for studying the properties of individual domains in a microphase separated system. A Voronoi tessellation provides the ability to measure domain volume, create an interfacial surface, and measure the coordination of spheres of the same domain. It can be used to simply measure the packing fraction and neighbors of particles modeled by multi-site spheres. We have also shown that the numerically efficient radical tessellation is an effective substitute for the Voronoi S tessellation for systems containing multiple sphere sizes. In Part III, Chapter 8, we found this measure to give critical insight as to the stability of the polydisperse TNS DG phase.

We also see potential for this method to provide other useful statistics for a soft matter system, such as studying the statistics of the Voronoi cells of the tethers to estimate configurational entropy, or measuring the net interfacial area, that we have not demonstrated here.

CHAPTER 10

Self Assembling Clusters Related to Mathematical Extremal Points on the Surface of a Sphere

This chapter corresponds to publication:
Phillips, C.L., Jankowski, E.R., Marval, M., Glotzer, S.C., Self Assembled Clusters of Spheres Related to Spherical Codes, Preprint

Anisotropic particles are compelling building blocks for self-assembled materials because their directional interactions can be exploited to create complicated and useful patterns[10,201–206]. One way to create anisotropic building blocks is to self-assemble them from simpler particles, where the building block represents a free-energy minimizing structure. Recently a number of papers have been published synthesizing and simulating compound building blocks that are clusters of sphere[2,205,207–212]. Colloidal spheres are attractive candidates for assembly because they can be made from a wide variety of polymers and metals, and their interaction potentials can be tuned with organic ligands, solvents, and salts.

Here we consider a class of self-limiting, or "terminal", colloidal clusters created by self-assembling a small population of one type of particle, the "halo" particle (HP), around a second type of particle, the central particle (CP). The clusters are terminal because the only attractive interaction is between the HP and CP, which are dilute in the fluid of HP, and therefore steric restrictions among co-adsorbed HPs inhibit further growth. The resulting clusters have structures determined by the interactions

among the adsorbed HPs, which self-organize around the CP to minimize their free energy.

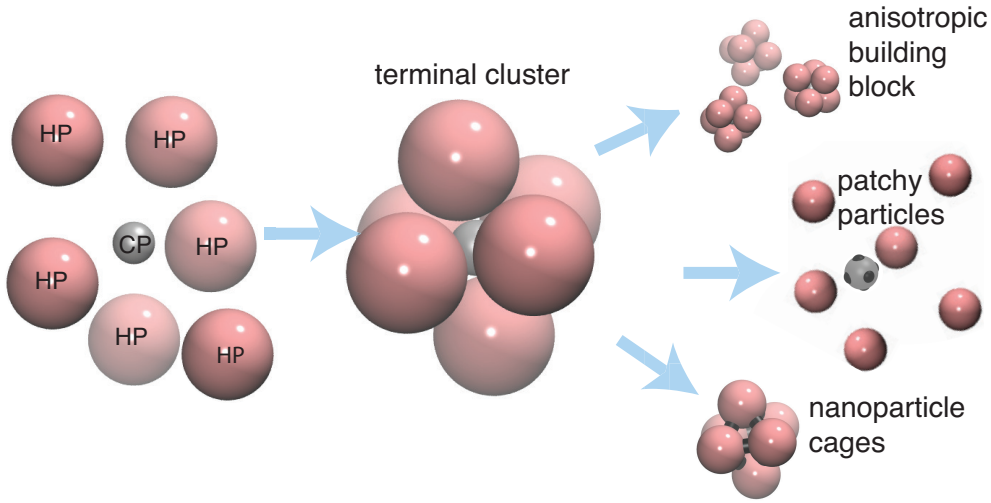


Figure 10.1: A terminal N-cluster with an octahedral structure ($N = 6$) is self-assembled from a bath of HP and a CP. This cluster has applications as a anisotropic building block, could be used to manufacture a "patchy particle" by imparting patches on the CP at the contact points, or could be locked into a nanocolloidal cage structure.

Arrangements of HPs on the surface of a CP have been studied extensively by mathematicians in the context of optimal arrangements of points on a sphere[38,213–215]. The solutions provide a library of anisotropic clusters that can in principle be created with properly designed interactions among the constituent particles. In this work we study hard sphere HPs that are attractive only to dilute CPs and not to other HPs, thereby producing clusters of HPs around a single CP. The arrangements of these HPs bear comparison to a particular set of solutions, the *spherical codes*, for certain ratios of particle diameters. We investigate the self-assembly of these clusters as a function of temperature, where entropy controls the equilibrium structure of the cluster, and in a semi-open system, where HP are free to bind and unbind from the CP surface. We also consider the effect of temperature on the cluster structures and dynamics at deviations from the perfectly dense packings that correspond to solutions of the spherical code.

This paper is organized as follows. In Section 1 we briefly review sphere surface extremal point problems. In Section 2, we introduce the methods we use to study the terminal N-clusters, including Brownian dynamics simulations, free energy calculations, and metrics for cluster structure and mobility. In Section 3, we report the results of our simulations, free energy calculations, and analyses. We find that terminal N-clusters self assemble across a range of diameters and temperatures and the structure of these clusters resemble spherical code solutions. These findings are supported by free energy calculations, which predict cluster sizes and distributions. Using Brownian dynamics and free energy calculations, we explain the surprising observation of a dominant low-symmetry $N = 5$ cluster, a deviation from the spherical code prediction. We calculate changes in cluster structure across a range of diameter ratios and investigate the dynamics for different cluster sizes, including collective modes. We find that the dynamics for clusters of different sizes are different. In Section 4, we discuss several ways this work can be extended to create more types of anisotropic particles via tuning of the particle interactions, constructing additional shells of particles, and creating structurally reconfigurable particles. In Section 5 we conclude with a summary of our findings.

10.1 Sphere Surface Extremal Points and Spherical Codes

The problem of finding extremal points obtained by optimally distributing points on the surface of a sphere to minimize a function f has been well studied in the field of mathematics[214–216]. The problem is typically posed as follows:

Given N points on the surface of a sphere of radius R, what arrangement of the N points minimizes a function f?

If $f = k\sum_{i\neq j}^{N} r_{i,j}^{-n}$, where $r_{i,j}$ is the Euclidean distance between the points i and j, and $n = 1$, minimizing f corresponds to the Thomson problem, whose solution describes the distribution of identical point charges on the surface of a sphere. As

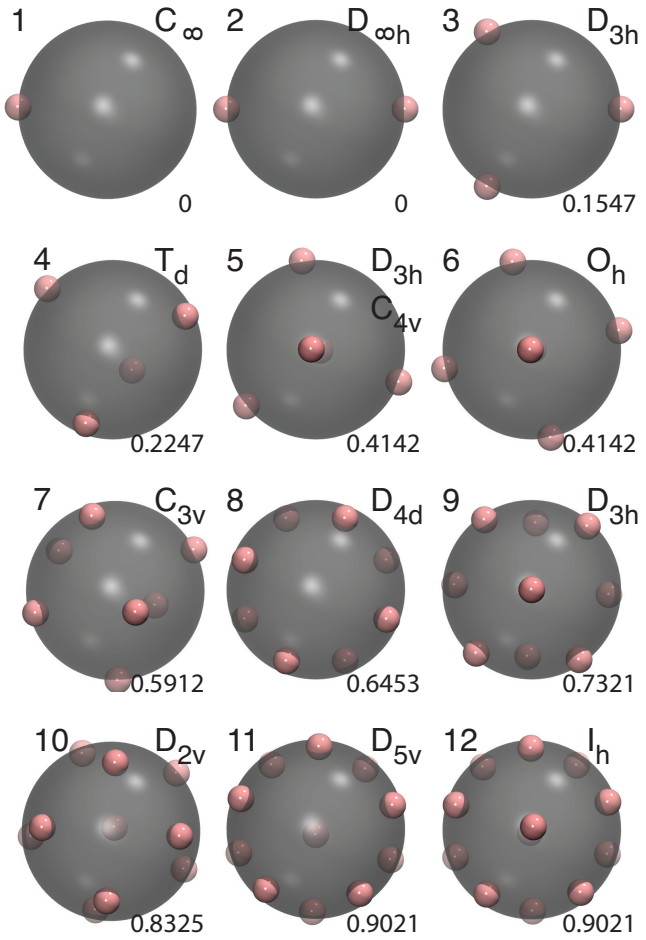


Figure 10.2: The arrangement of points (pink) that correspond to each spherical code solution for $1 \leq N \leq 12$. The point group of each arrangement is shown to the upper right of each arrangement, and the densest packing diameter ratio $D_c/D_h = \Lambda_N$ is shown to the lower right. For $N = 5$, the triangular bipyramid configuration is shown. Other $N = 5$ configurations are shown and discussed in Figures 10.10-10.9 .

$n \to \infty$, the problem corresponds to the *spherical code*, (also known as the Fejes Tóth, or Tammes problem), whose solution maximizes the minimum distance between any two sets of points[38,213–215]. Other possible choices for f include minimizing the maximum distance of any point to its closest neighbor, also known as the *sphere covering problem*, and maximizing the volume of the convex hull of the points. For each of these problems solutions are exactly known for some values of N, while various numerical searches have suggested best solutions for other N. For the functions

mentioned, tables of putative solutions up to at least $N = 130$ can be found in Ref.[38].

Fig. 10.2 depicts the spherical code solutions for $1 \leq N \leq 12$. The arrangement of points for $N = 4$ corresponds to the vertices of a regular tetrahedron, $N = 6$ an octahedron, $N = 8$ a square anti-prism, and $N = 12$ an icosahedron. The point arrangement of $N = 11$ is equal to the $N = 12$ solution minus a single point, or an icosahedron with one truncated pentagonal face. For each N, the point group – the group of isometries that keeps one point fixed – of the arrangement[214] is shown in the upper right corner. Each optimal arrangement of N points on the surface of the sphere is unique except for $N = 5$ which has a continuum of solutions ranging from a triangular bipyramid (point group D_{3h}, shown in Fig. 10.2 and Fig. 10.10b) to a square pyramid (point group C_{4v}, shown in Fig. 10.10a). All solutions in the continuum have two points at opposite poles of the central sphere and differ by the positions of the three remaining points on the equator. The square pyramid arrangement is equal to the $N = 6$ solution minus a single point. We discuss these structures in detail in Chapter 10.3.4.

If the N points represent sphere centers, the spherical code solution corresponds to the densest packing of N hard halo spheres that all "kiss" a central sphere. For any packing of spheres around a central sphere, we define Λ to be the ratio of the central sphere diameter, D_c, to the halo sphere diameter, D_h. We denote the minimal possible diameter ratio for N spheres, which corresponds to the spherical code solution, as Λ_N. In Fig. 10.2, Λ_N of each arrangement is shown to four significant digits in the bottom right corner. Notably, $\Lambda_{N=5} = \Lambda_{N=6}$ and $\Lambda_{N=11} = \Lambda_{N=12}$. In one of mathematics' most famous debates, Isaac Newton and David Gregory argued whether the kissing number of unit spheres ($\Lambda = 1$) is 12 or 13. Had it been known that a central unit sphere can only be kissed by 13 spheres if their radii is $r \leq 0.9165$, or $\Lambda_{13} = 1.0911$[38], this would have settled the question. Isaac Newton's conjecture that the kissing number is 12 was not proven until 1953[197].

We also note that the spherical code solutions for $N = 3$-12, except $N = 5$, are rigid or jammed. They contain no "rattlers", defined as spheres not in isostatic contact with other spheres[217,218], and cannot be deformed other than global isometries[217,218].

10.2 Methods

To predict and compare the terminal N-clusters of halo particles bonded to central particles we use computational tools that sample equilibrium statistical mechanical ensembles. In particular, Brownian dynamics simulations of model particles are used to perform computer experiments of self-assembly and the results of these simulations are compared against cluster probabilities calculated from a free energy analysis based upon numerical partition function calculations[206]. We also calculate detailed structural and dynamic quantities for each cluster.

10.2.1 Hard Sphere and Sticky Sphere Model

In a semi-open system, the spherical code solutions of Section II correspond to perfectly hard spheres adsorbed on a perfectly sticky sphere. Mathematically, perfectly hard spheres are points interacting via a function that steps from infinity to zero (Fig. 10.3a) and perfectly sticky spheres are points interacting via the same function plus an infinitely narrow square well function (Fig. 10.3b). In this work we use radially-shifted Weeks-Chandler-Andersen (WCA) and Morse models of hard and sticky spheres, respectively, which allow for computational efficiency as well as direct comparison with their ideal mathematical counterparts (Fig 10.3). They also capture, in a general sense, the repulsive and attractive interactions of the constituent particles we have in mind. As nanoparticle synthesis continues to mature, the types of interactions that can be used to guide the self-assembly of small particles can be precisely tuned over wide ranges of length and energy scales, and the models used in simulations can be suitably adjusted.

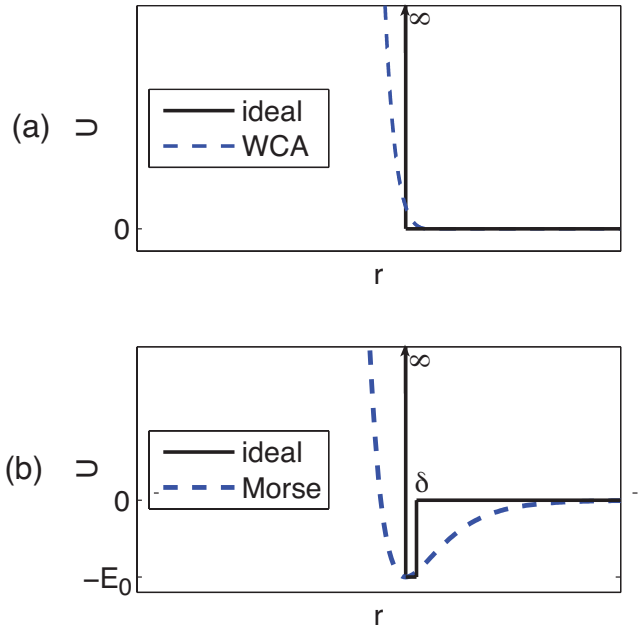


Figure 10.3: (a) A mathematically ideal hard particle interaction is shown in solid black compared to the hard particle interaction (in dashed blue) given by the WCA potential (Eqn. 2.6). (b) A sticky sphere with a kissing contact potential when $\delta \to 0$ is compared to a model sticky sphere (in dashed blue) given by the Morse potential (Eqn. 2.7).

The radially-shifted WCA potential is given by[42] Eqn. 2.6 of Chapter 2.2.3. The shifting parameter α is defined as $\alpha = \sigma_h - \sigma$, where σ_h is the WCA "diameter" of the HP, and $r_{cutoff} = 2^{1/6}\sigma + \alpha$. The interaction between two HPs can be made arbitrarily hard relative to their size by increasing σ_h. The cost of increasing σ_h is that the dimensionless time τ that elapses over each time step is reduced as $\tau \propto 1/\sigma_h$. We choose $\sigma_h = 3\sigma$ for its computational efficiency and its relatively "hard" modeling of spheres. The energy parameter ϵ also determines the "hardness" of the HP, as a larger energetic penalty to overlapping corresponds to a "harder" potential. The cost of increasing ϵ is that a smaller simulation time step is needed to model a steeper function. The energy parameter is set to $\epsilon = (0.1/T)$, where T is the temperature of the simulation, so that the hardness of the HP is independent of temperature.

Using Eqn. 2.6 to model hard HPs is effective because of the large potential energy penalty associated with two HPs approaching closer than 3σ. However, due to the soft nature of the potential, spheres with sufficient kinetic energy can, in principle, approach as close as 2σ. It is therefore useful to determine the effective hard particle diameter of HP modeled by Eqn. 2.6. We use the Barker-Henderson equation[219] to calculate the effective diameter

$$D_{h,e} = \int_0^{\sigma_{BH}} (1 - e^{-\beta u(r)}) dr \simeq 3.0786\sigma \tag{10.1}$$

where $\beta = 1/k_B T$, $u(r) = U_{WCA}$ and the potential is zero at $\sigma_{BH} = 2^{1/6}\sigma + \alpha$. For the purpose of assessing the error in our calculations based on the effective diameter, we can characterize two HPs as contacting when the interaction energy between them is in the range $0 < U_{WCA} < 10k_B T$, which corresponds to $3.0\sigma < D_{h,e} < 3.1225\sigma$.

The radially-shifted Morse potential[43], Eqn. 2.7 of Chapter 2.2.4, used to model the "kissing" contact potential between the HP and CP. The parameters of the potential are $E_0 = 5$ which determines the depth of the energy well, $\beta = 5.0/\sigma$ which determines the width of the energy well, and r_0 which determines the radial displacement of the bottom of the energy well. The Morse potential interaction range is truncated at $r_{cutoff} = 2.5\sigma + (r_0 - \sigma)$. An effective CP diameter can be calculated by defining an HP and CP as *bonded* when the distance between the two particle centers is at the minimum of Eqn. 2.7. More properly, an HP and CP are bonded when they remain positionally correlated because the HP remains within a given displacement of the CP. We use the minima of Eqn. 2.7 and the effective diameter of Eqn. 10.1 to define an effective CP diameter $D_{c,e} = 2r_0 - D_{h,e}$. The ratio of the CP to HP diameter is therefore $\Lambda^m = D_{c,e}/D_{h,e}$, ($m$ for *molecular dynamics*). We choose to keep the hardness of the HP-HP interaction constant for all the MD simulations by holding $D_{h,e}$ (i.e. σ_h) constant while varying r_0 to change $D_{c,e}$.

The non-infinitesimal potential well width of Eqn. 2.7 permits the bond between the CP and HPs to stretch a small amount while remaining bonded. At some Λ^m ratios, this stretching, though small, may be enough to accommodate an additional HP bond to the CP. It is therefore useful to define the CP *bond-stretched effective diameter* $D_{bs,c,e} = 2\bar{R}_{hc} - D_{h,e}$, where $\bar{R}_{hc}$ is the average center-to-center distance between a bonded HP and CP measured in a simulation. The bond-stretched diameter ratio is defined as $\Lambda^{bs} = D_{bs,c,e}/D_{h,e}$ (*bs* for *bond-stretched*). Λ^{bs} is always greater than Λ^m. When the cluster is loosely packed, the difference between the two measures converges to zero.

10.2.2 Brownian Dynamics

To model mixtures of halo particles and central particles assembling in a thermal bath we perform Brownian dynamics (BD) simulations, implemented in HOOMD-blue[73]. The natural units of this system are: the effective diameter of the HP, $D_{h,e} = 3.0786\sigma$; the mass of a HP, m; and the depth of the HP-CP energy well, E_0. The volume fraction, ϕ, is defined as the ratio of the total volume of the HPs and CPs to the simulation box volume, the dimensionless time is $t^* = t/(D_{h,e}\sqrt{m/E_0})$, and the dimensionless temperature is $T^* = k_BT/E_0$. We use periodic boundary conditions. Each particle is subjected to conservative, random, and drag forces, and its motion is governed by the Langevin equation discussed further in[24,122,155]. We use a value for the drag coefficient $\gamma = 0.726\ m/t^*$. The same drag coefficient is applied to HPs in both the free and bound state. The conserved forces between particles are per Eqns. 2.6 and 2.7 above.

10.2.3 Free Energy Calculations

The relative probability of finding a particular cluster of N HPs bound to a CP can be predicted using free energy calculations detailed in references[206,220,221]. For a

given Λ, the partition function is defined by the appropriately weighted sum over all possible configurations of N HPs bound to a CP for $N = 1$ to ∞. The contributions of the distinguishable microstates to the partition function are calculated numerically.

The partition function is calculated assuming ideal hard spheres and sticky spheres (Fig. 10.3). Given HPs of diameter D_h and a CP of diameter D_c, the interaction potential between ideal HPs is defined as,

$$U_{H-H}(r) = \begin{cases} \infty & r < D_h \\ 0 & r \geq D_h \end{cases} \tag{10.2}$$

and the interaction potential between an ideal HP and an ideal CP is defined as

$$U_{H-C}(r) = \begin{cases} \infty & r < (D_c + D_h)/2 \\ -E_0 & r = (D_c + D_h)/2 \\ 0 & r > (D_c + D_h)/2 \end{cases} \tag{10.3}$$

We define $\Lambda^f = D_c/D_h$ (f for *free energy calculation*). As in Chapter 10.2.1, to vary Λ^f, D_h is held constant (set to $D_{h,e}$ from Eqn. 10.1) and D_c is changed.

If $\Lambda_{N=M} \leq \Lambda^f < \Lambda_{N=M+1}$, then configurations of M HPs bonded to the CP minimize the potential energy and configurations with more than M HPs have infinite potential energy (zero probability). Configurations with fewer than M HPs bonded to the CP increase the entropy of the cluster. When $k_B T \approx E_0$, the free energy can be minimized by clusters with fewer than M HPs, because the entropy gained by the remaining HPs on the CP balances the increase in potential energy. In the grand canonical ensemble, at a fixed Λ^f, the probability of observing a particular cluster s is given by the Boltzmann distribution:

$$P_s = e^{-\beta F_s} = \frac{\Omega_s e^{-\beta(U_s - \mu N)}}{\mathcal{Z}} \tag{10.4}$$

where $\mathcal{Z} = \sum_s \Omega_s \exp(-\beta(U_s - \mu N))$ is the partition function and $U_s - \mu N = NE_0$. Without loss of generality, we treat μ =0. (A non-zero μ will only induce a uniform temperature shift in our final results.)

In practice, calculating $\mathcal{Z}$ exactly is difficult, but by assuming that only a small number of clusters contribute to $\mathcal{Z}$ [206,220,221], the relative probabilities of these clusters can be determined. As in [206,220,221], the degeneracy Ω_s can be written as a product of three independent terms, the translational, Z_t, rotational, Z_r, and vibrational Z_v partition functions. The translational partition function is approximately equal for all the clusters because they are all small compared to the accessible volume, and thus contributes equally to the Ω_s of each cluster.

To calculate the rotational and vibrational partition functions for an N-cluster, we first assume an equilibrium configuration defined by N HPs in a spherical code configuration at a radial displacement of $(D_c + D_h)/2$ from the CP. The rotational partition function is then calculated as $Z_r = c_r \frac{\sqrt{I}}{\kappa}$, where c_r is a temperature-dependent constant that is the same for all the clusters, I is the determinant of the moment of inertia tensor, and κ is the symmetry number of the spherical code configuration under rotation. Each sphere is given a unit mass. The vibrational partition function is proportional to the product of the vibrational freedom, or freedom to rattle, of each sphere in the cluster. The vibrational freedom of each HP can be measured as the fractional area of the surface of the CP it has access to, subject to the restrictions imposed by its neighboring spheres. We approximate the vibrational area available to a given HP in a particular configuration by using a Monte Carlo numerical approach whereby new positions for the HP are randomly generated and accepted if the HP does not overlap another HP. The accessible vibrational area is proportional to the total number of accepted positions that are part of a contiguous area that includes the HP's original position divided by the total number of random trials. If $\Lambda^f = \Lambda_N$, when the diameter ratio matches the spherical code ratio, then most, if not all, of the

spheres in an N-cluster are jammed and have no vibrational freedom.

The free energy calculation is approximate, as it does not consider the contribution of collective modes of HP motion to $\mathcal{Z}$, which, in certain systems can help stabilize one configuration over another[222]. As we show in Chapter 10.3.3, each cluster has a small Λ range, $\Lambda > \Lambda_N$ where collective modes are not present, and only local rattling is observed. Outside this range, we expect some error in the calculation of relative probabilities to accumulate. The benefit of this free energy approximation is demonstrated by both its favorable comparison to predictions made by BD simulations and by its ability to rapidly predict the entire phase diagram. Applying a more computationally intensive method to perform an exact free energy comparison would be an interesting topic for future study.

10.2.4 Structure and mobility measures of a cluster

The HPs in an N-cluster for $\Lambda = \Lambda_N$ are confined to a unique N spherical code solution and cannot rearrange or even rattle for $N = 3$ to 12, excepting $N = 5$. For $\Lambda >> \Lambda_N$, the HPs are free to randomly arrange on the surface of the CP. We aim to understand the structure and dynamics of the N-cluster between these extremes. We perform BD simulations of pre-assembled clusters wherein the HPs are restricted to the surface of the CPs, a constraint imposed during the integration of the equations of motion. This allows the dynamics of HP rearrangement to be isolated from the dynamics of assembly and disassembly and prevents any stretching of bonds from influencing the structures observed. Similar to Chapter 10.2.1 the CP diameter is defined as $D_{c,e} = 2r_0 - D_{h,e}$, where r_0 is the fixed distance between the CP and HP centers and $D_{h,e}$ is the same as defined in Eqn. 10.1. The CP to HP diameter ratio in the constrained system is thus $\Lambda^c = D_{c,e}/D_{h,e}$ (c for *constrained*) and varied by changing r_0. Λ_c is initialized such that the HP can be sparsely randomly distributed on the surface (i.e. $\Lambda^c >> \Lambda_N$), and then slowly decreased over the course of a

simulation to a target Λ^c.

The angular displacement between two HP bound to the same CP, or $\theta = \angle ACB$ is defined by the centers of two HPs A and B and the CP C. To characterize the structure of the cluster, the distribution of angular displacements between pairs of HPs, $n(\theta)$ for $\theta = [0, \pi]$ are measured for a fixed N and Λ^c over all HP pairs every 10^4 time steps during a simulation with 10^9 total time steps. The value of $n(\theta)$ for a given N and Λ^c represents the likelihood of finding an HP at an angle θ relative to a given HP, and $\int n(\theta)d\theta = N - 1$. $n(\theta)$ is analogous to a pair correlation function.

To characterize the dynamics we calculate the time scale over which θ is no longer correlated with itself. We define the mobility parameter τ from

$$C(\theta(t), \theta(t + \delta t)) = e^{-\tau t} \tag{10.5}$$

where $C(\theta(t), \theta(t + \delta t))$ is the normalized angular autocorrelation function and t is time. In this work, τ has units of 1/10,000 time steps. The more mobile an HP is on the surface of the CP, the more rapidly its angular displacement with respect to other HP decorrelates. When the rate of decay of angular correlations is zero, all the HP in the cluster are fully caged. We only calculate τ for clusters that display more than one distinct peak in their $n(\theta)$ distributions so that position swapping can be distinguished from local rattling. The lower bound on the τ measurement is $1.5 \cdot 10^{-4}$ because below this the HP position swaps occur too infrequently over a 10^9 time step simulation for accurate values of τ to be measured. We calculate τ as a function of N and $\Delta\Lambda^c = \Lambda^c - \Lambda_N$, or the difference between the diameter ratio of the N-cluster and the N spherical code solution ratio. Because the HPs in the simulation are not perfectly hard and are constrained to the CP surface, it is possible for meaningful measurements to be made when $\Delta\Lambda^c < 0$.

10.2.5 The calculation of Λ

In this paper, to elucidate different properties of N-clusters, several calculation methods are used, necessitating four different ways to determine $\Lambda = D_c/D_h$, the ratio of the central particle diameter, D_c, to halo particle diameter, D_h. Each way was chosen to best represent the effective diameters of the HP and CP in the particular method. These Λs are comparable to each other and to the spherical code solution ratios, Λ_N of Fig. 10.2. We indicate the calculate type by the superscript x of Λ^x, where $x \in \{m, bs, c, f\}$, where the m, Brownian (molecular) dynamics; bs, bond-stretched; c, constrained; and f, free energy calculation, as defined above.

10.3 Results

10.3.1 Self-assembly and free energy of N-clusters

Using Brownian dynamics we simulate the self-assembly of clusters as a function Λ^m and at two different temperatures to investigate the effect of thermal noise on the distribution of stable terminal N-clusters. We compare these results to the known spherical code solutions and to free energy calculations.

Brownian dynamics simulations of self-assembly are initialized by placing 1000 CPs on a cubic lattice, spaced so as to behave as independent systems. The lattice is embedded in a bath of HPs at a total volume fraction of $\phi = 0.24 - 0.27$. The bath contains a minimum of four times as many HPs per CP as the maximum cluster size observed for that Λ^m. We perform a total of 760 simulations of $20{\times}10^6$ time steps, with time step size $\Delta t^* = 0.00363$ at low ($T^* = 0.02$) and high ($T^* = 0.1$) temperatures. With this set of simulations, we calculate the cluster size distribution as a function of Λ^m.

At the low temperature we observe that the cluster sizes are highly monodisperse as a function of Λ^m. In Fig. 10.4, the mean cluster size assembled at the low tem-

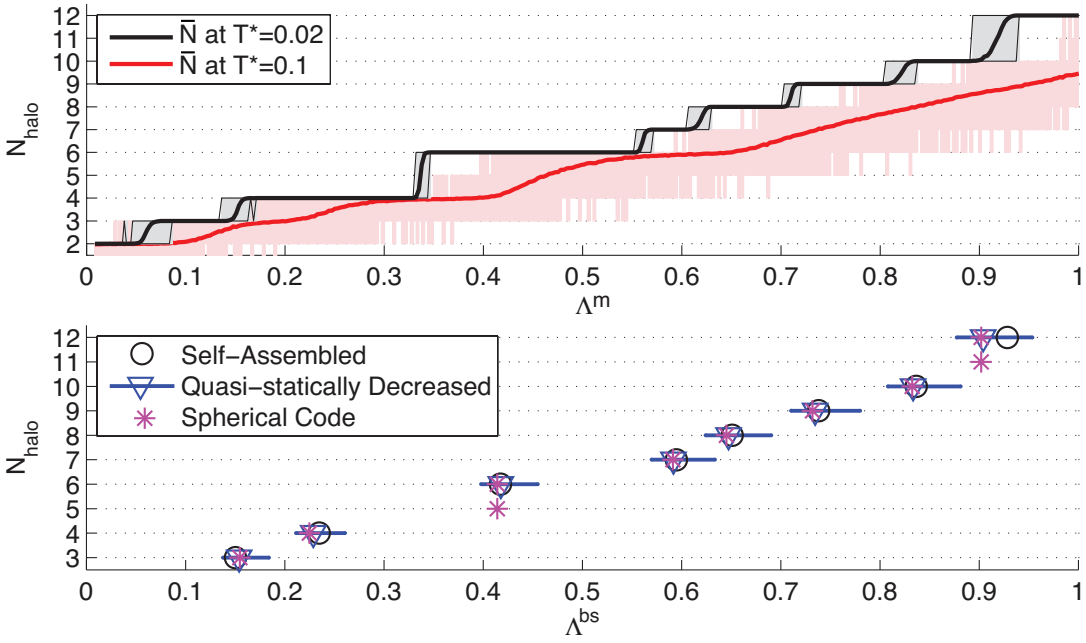


Figure 10.4: Top: The N clusters that self-assemble as a function of Λ^m and temperature is shown. The average N of the self-assembled cluster at $T^* = 0.02$ is shown as a black line. The maximum and minimum N in the simulation is shaded grey. The average N of the self-assembled cluster at $T^* = 0.1$ is a red solid line. The maximum and minimum N in the simulation is shaded pink. Bottom: Accounting for bond-stretching and the effective diameter of the HP, the lowest ratio where a cluster of size N observed in the quasi-statically decreasing simulation (blue triangles) and for the self-assembled simulations (black circles) are compared to the spherical code predictions (pink star). Error bars for the quasi-static simulation ratios are generated from the contact range of two HP.

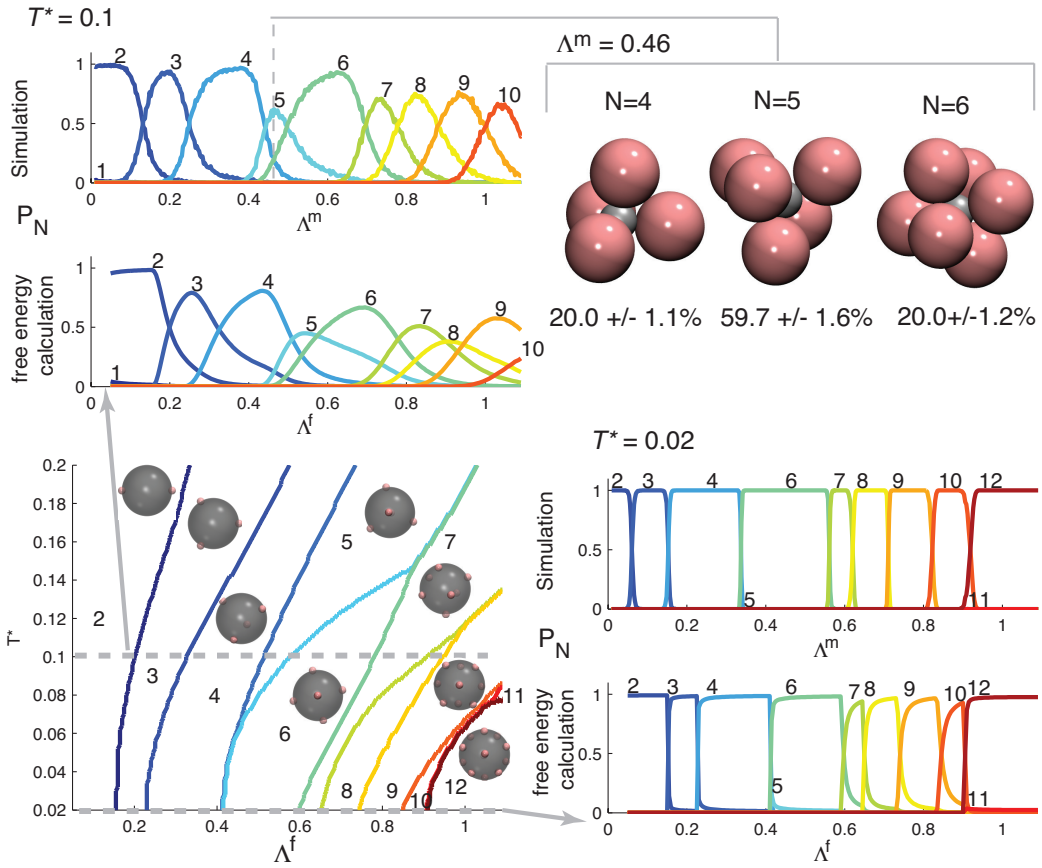


Figure 10.5: The distributions of cluster sizes as a function of temperature and Λ as given by the free energy calculation and the BD simulations are compared. Bottom left corner: phase diagram of the free energy prediction of the most probable cluster size. Lower right and upper left corners: in-page slices of the probability of finding each cluster size P_N as predicted by the free energy calculation and BD simulation at the high and low temperature. Upper right corner: the three most common clusters found in the BD simulation at the high temperature and $\Lambda^m = 0.46$.

perature, $T^* = 0.02$ for $0.01 < \Lambda^m < 1$ is shown as a black solid line. Grey shading indicates the range of cluster sizes observed at a particular Λ^m. Over this range of Λ^m clusters are uniform in size except when Λ^m is near a value where there is a transition from one mean cluster size to another. At these transitions, we observe a narrowly distributed mixture of cluster sizes; e.g., at $\Lambda^m = 0.71$ for the $T^* = 0.02$ curve, we find equal numbers of clusters containing 8 or 9 HPs.

In comparison to the low temperature data, the clusters at high temperature are both smaller on average, and have a broader distribution of sizes as a function of Λ^m. In Fig. 10.4, we show the distribution of clusters assembled at high temperature, $T^* = 0.1$ (red). The region shaded pink represents the range of cluster sizes measured at a given Λ^m at $T^* = 0.1$. At $\Lambda^m = 0.71$ for the $T^* = 0.1$ curve, we now observe clusters of 5, 6 7, and 8 HPs. We also observe that the $N = 5$ and $N = 11$ clusters are not stable at any Λ^m at low temperature but are present in the broader distribution of clusters at high temperature.

To test the stability of the self-assembled clusters at low temperature, we perform a simulation wherein the diameters of the CPs in large pre-assembled clusters are slowly decreased. A single system with $\Lambda^m = 0.9489$ is equilibrated for $20{\times}10^6$ time steps at $T^* = 0.02$, at which time every CP is bonded to 12 HPs. Subsequently $D_{c,e}$ is decreased at a rate of $4.833 \times 10^{-8}\sigma/\Delta t$ until $\Lambda^m = 0.0101$. As discussed in Chapter 8.1.3, this decrease in the diameter is slow enough that the system remains quasi-static, i.e. the system is approximately in equilibrium. For this system, as Λ^m approaches a transition ratio, 1-3 HPs detach from a given CP and re-enter the bath, until only two HP are bonded to each CP. In effect, this quasi-statically decreased $D_{c,e}$ simulation disassembles the clusters as a function of Λ^m.

If bond-stretching is taken into account, we find that at a low temperature ($T^* = 0.02$) the N-clusters self-assemble at the Λ ratio predicted by the spherical code solutions. In tightly packed clusters, bond stretching makes $\Lambda^{bs} > \Lambda^m$. In the bottom

plot of Fig. 10.4, the lowest Λ^{bs} at which a cluster of size N is observed for the self-assembled (black circles) and quasi-statically decreased (blue triangles) simulation data is shown and compared to the spherical code Λ_N ratio (pink stars). Blue error bars indicate the Λ^{bs} ranges from quasi-statically decreased simulations, generated by assuming that the true diameter of an HP is the limits of the contact range defined in Chapter 10.2.1. Good correspondence between the predicted and measured ratios is observed when bond-stretching and the appropriate effective diameters of the particles is accounted for.

We calculate the free energies of all clusters from $N = 2$ to $N = 12$ over a temperature range of $0.02 \leq T^* \leq 0.2$ and diameter ratio range of $0.05 \leq \Lambda^f \leq 1.09$. In the bottom left plot of Fig. 10.5, we report a "phase diagram" of the most probable cluster at each combination of T^* and Λ^f. The plots in the bottom right and top left of Fig. 10.5 show data from an in-page slice of the phase diagram at the low and high temperature, $T^* = 0.02$ and $T^* = 0.1$, and directly compare it to cluster distributions from the BD simulation data of Fig. 10.4. For example, the three clusters and probabilities depicted in the upper right of Fig. 10.5 are from a single high temperature BD simulation with $\Lambda^m = 0.46$.

We see that the free energy calculations support the findings of the BD simulations. At high temperature and at a given Λ, both show a decrease in cluster size relative to the low temperature data, as well as a broadening in the distribution of cluster sizes. Discrepancies in peak height and shape between the two predictions in Fig. 10.5 are likely due to the soft sphere approximation, not accounting for the change in the contact energy or effective diameter due to bond-stretching, and also to neglecting the collective vibrational modes in the free energy calculations. However, the free energy calculation shows that most of the features of the BD simulations at higher temperatures can be attributed to the offsetting of the increase in potential energy (i.e. fewer bonded HPs) by the commensurate increase in vibrational freedom of the

remaining bonded HPs.

Consistent with the BD simulation data, the free energy calculation also predicts that $N = 5$ clusters are not stable at low ($T^* < 0.06$) temperatures. Spherical code solutions indicate that the densest $N = 5$ clusters occur at the same Λ as the densest $N = 6$ cluster. Thus, at low temperature, when the free energy is dominated by the potential energy term, the $N = 6$ cluster is always stable over an $N = 5$ cluster. However, the free energy calculation predicts that at $T^* > 0.06$ there is a Λ range where an $N = 5$ cluster is the most probable cluster. This Λ range is observable in the high temperature BD simulation data. The stabilization of the $N = 5$ cluster over the $N = 6$ cluster at higher T^* arises from the non-negligible contribution of the vibrational partition function, the only term in the partition function that significantly differs between the two clusters. The $N = 11$ cluster is similarly predicted to be unstable at low temperatures but stable over the $N = 12$ cluster at a higher temperature. The free energy calculation also predicts an entropic stabilization of the $N = 7$ and 9 clusters over the $N = 8$ and 10 clusters, respectively, at higher T^* and "triple points", at which the probabilities of three clusters (e.g. 4, 5, and 6; or 7, 8, and 9) are equal.

10.3.2 Structure of N-clusters

We next consider how the structure of each N-cluster changes as $\Lambda > \Lambda_N$. Clusters that have a large range of Λ over which their structure is ordered and stable are desirable targets for synthesis. We investigate the structure and dynamics of the clusters by modeling HPs constrained to the surface of a CP, as described in Chapter 10.2.4 for different N and Λ^c.

In simulation, we observe that a cluster of size N generally exhibits three different dynamics over different ranges of Λ^c. In the first range, each HP remains locally caged. Each HP of the N-cluster can be assigned to one point of the N spherical

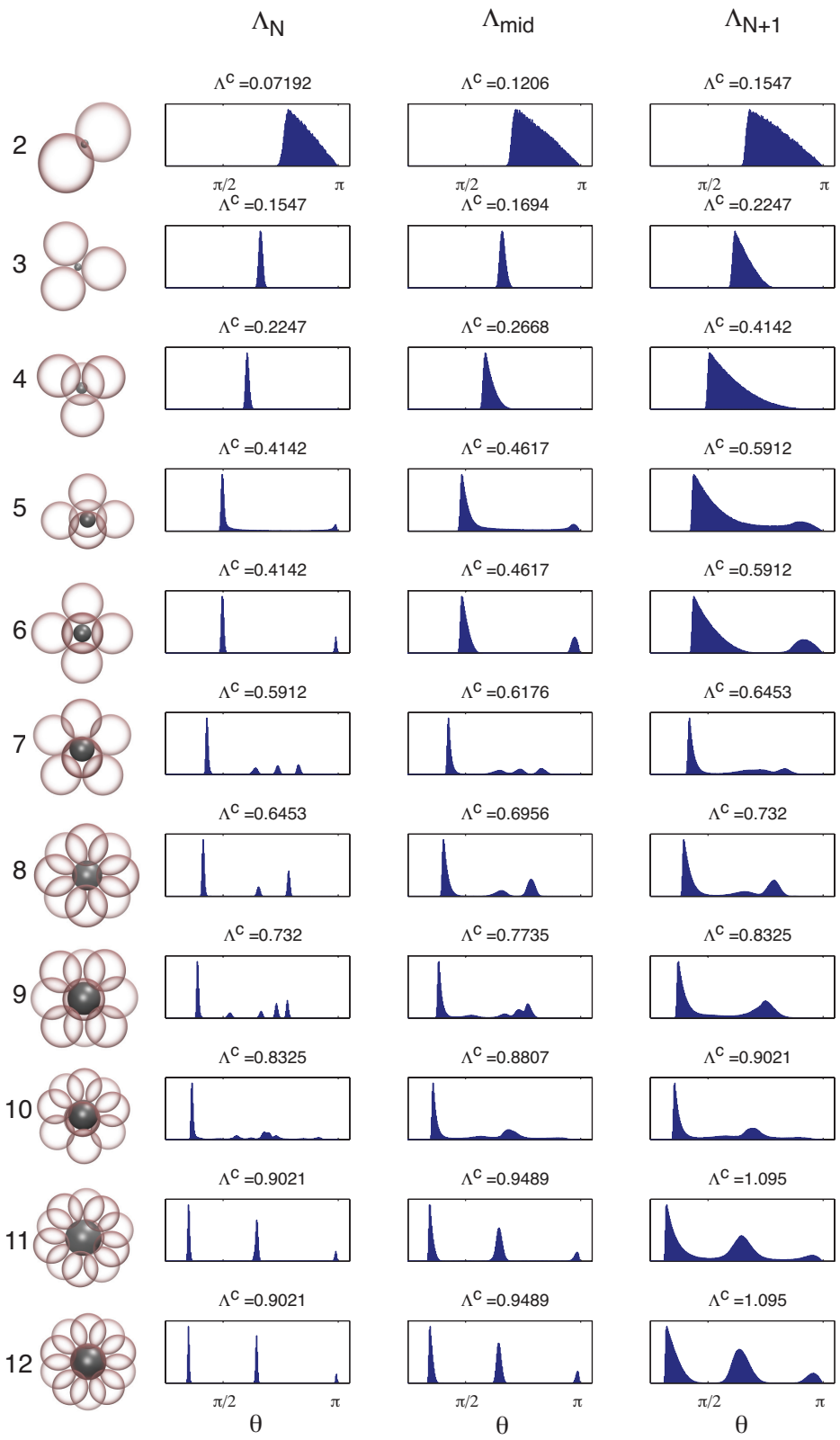


Figure 10.6: The distribution of angular displacements $n(\theta)$ for each cluster. The $n(\theta)$ shows a structural fingerprint particular to each cluster.

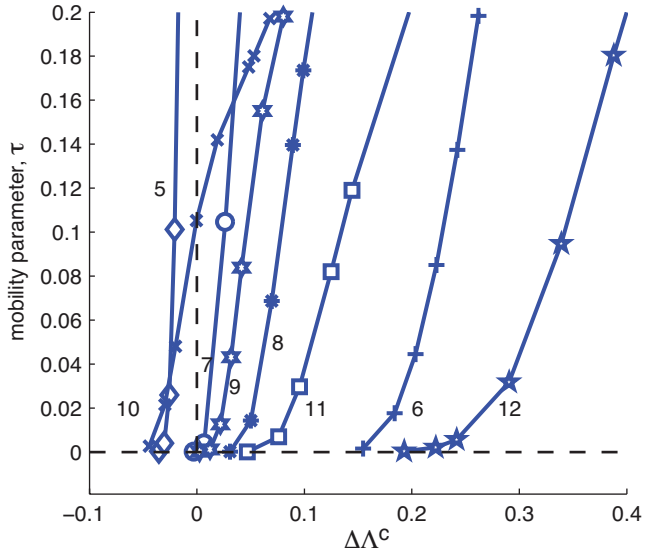


Figure 10.7: Cluster mobility as a function of the $\Delta\Lambda^c = \Lambda^c - \Lambda_N$. Note that for the $N = 6$ and $N = 12$ clusters, the HPs do not become measurably mobile for $\Delta\Lambda^c >> 0$. At the other extreme, $N = 5$ and $N = 10$ are mobile for $\Delta\Lambda^c < 0$.

code solution of Fig. 10.2 and that mapping remains invariant under the dynamics of the cluster. Like an atom in a crystal lattice, each HP rattles about its point. In the second range, each HP can almost always be assigned to one point of the spherical code solution, however the mapping does not remain invariant under the dynamics of the cluster. The HPs sporadically rearrange but are still generally found rattling about the points of the spherical code solution. In the third range, the HPs cannot be assigned to points of the spherical code solution and move freely on the surface. Between the second and third range, we suspect that there is no distinct measurable boundary, but simply an increasing likeliness of a cluster being in "transitional" states. Below we show how the two measures introduced in Chaper 10.2.4 capture the signature features of these ranges.

In Figure 10.6, the distribution of angular displacements, $n(\theta)$, is shown for $2 \leq N \leq 12$ HPs constrained to the surface of a CP at $T^* = 0.02$. For each N, $n(\theta)$, is shown for three different Λ^c, corresponding to Λ_N, Λ_{N+1} and a midpoint between the

two. These three distributions are shown in order of increasing Λ^c from left to right. Note that for $N = 2$, $\Lambda_{N=2}$ is zero, as it is always possible to add a second HP to a CP with one bound HP, regardless of CP size. In this case, we arbitrarily choose the minimum $\Lambda^c = \Lambda_{N+1}/2$.

For each cluster we observe a unique $n(\theta)$ structure fingerprint that softens as Λ^c increases. For $N \leq 4$, each HP has one equidistant ring of neighbors, resulting in $n(\theta)$ having a single peak that broadens as Λ^c increases. For $N > 4$, each $n(\theta)$ has multiple peaks. For $N > 4$ except $N = 5$ and $N = 10$, the first peak at Λ_N is narrow and not connected to other peaks, indicating HPs are locally caged at their spherical code points[223]. The width of a peak is proportional to the rattling of a HP within its local cage. As Λ^c increases, the peaks broaden and eventually become connected. This broadening and overlapping is associated with the degradation of the well-defined structure by increased rattling and sporadic rearranging. In no case did we find any evidence of new structures emerging. We note that for the clusters $N = 5$ and $N = 10$ the peaks are not distinct at the smallest Λ^c considered. For all N, if $\Lambda^c >> \Lambda_N$, then the HPs sample uniformly random arrangements on the CP surface and the $n(\theta)$ distribution is a cosine function of θ, truncated to zero when θ is less than the angular diameter of the HP (e.g. in Fig. 10.6, $N = 2$, $n(\theta)$ is a truncated cosine function for each Λ^c). In Fig. 10.6 for $N > 2$, insofar as the distributions are far from converged to a cosine function, we observe structure derived from the underlying spherical code solution over the entire range of Λ considered

10.3.3 Mobility of N-clusters

We next consider the dynamics of the HPs on the CP surface. As described in Chapter 10.2.4, we can measure how rapidly the angular displacements of the HPs decorrelate at a given Λ^c. We call this measure the mobility parameter, τ. When $\tau = 0$, the HPs are in the first dynamical range; that is, each HP is fully caged and the

angular displacement between any two HPs does not decorrelate. When $\tau > 0$, but small, the HPs are in the second dynamical range. In Figure 10.7, the τ of different clusters are calculated as a function of increasing Λ^c relative to the ratio at which the cluster is predicted to assemble, $\Delta\Lambda^c = \Lambda^c - \Lambda_N$. We observe that the size of the first dynamical range varies widely among clusters. Noticeably, the $N = 6$ and $N = 12$ clusters are not measurably mobile until $\Delta\Lambda^c$ is large. Note that in Fig. 10.6, the midpoint $n(\theta)$ data of both $N = 6$ and $N = 12$ still have distinct separated peaks. In contrast, the $N = 11$ cluster becomes mobile at a much lower $\Delta\Lambda^c$ than $N = 12$ cluster, despite having nearly the same spherical code solution and $\Lambda_{N=11} = \Lambda_{N=12}$. Comparing the $N = 11$ and $N = 12$ distributions in Fig. 10.6, at $\Lambda^c = 0.9021$ the two clusters have nearly identical $n(\theta)$ distributions. At $\Lambda^c = 0.9489$, $N = 11$ is measurably mobile ($\tau = 1.5 \cdot 10^{-4}$)) but still has distinct peaks in $n(\theta)$ that are only slightly softer than that of $N = 12$. By $\Lambda^c = 1.095$ the $n(\theta)$ peaks are noticeably softened for the $N = 11$ cluster relative to that of the $N = 12$ cluster and the peaks are connected. At this ratio, the mobility of the $N = 11$ cluster is $\tau = 0.19$ while the $N = 12$ cluster is just measurably mobile ($\tau = 4 \cdot 10^{-4}$). In comparison, the $N = 10$ cluster is mobile even at the ratio at which it first self-assembles. The rapid increase of the mobility as a function of increasing Λ^c in Fig. 10.7 is consistent with the connected peaks and the rapid softening of the peak structure for $N = 10$ in Fig. 10.6.

The fact that HP mobilities for a particular cluster depend upon the cluster's structure is not surprising. However, it is not obvious that clusters of different sizes should have such variation in the widths of the first dynamical range indicated in Fig. 10.7. There is little correlation, for example, between the mobility of the N cluster and the range of Λ over which the N cluster is stable in Fig. 10.4. We find that the HPs in the $N = 5$ and $N = 10$ clusters are never fully caged, while the HPs for $N = 6$ and $N = 12$ are fully caged for a large range of Λ^c. The mobilities for $N = 7$,

8, 9, and 11 lie between these extremes. Note that $N = 6$ and $N = 12$ clusters have highly symmetrical spherical code point arrangements with octahedral and icosahedral structures, respectively. Their HP centers define the vertices of Platonic solids with equilateral triangle faces. For N = 7, 8, 9, 10, and 11, the convex polyhedra defined by the centers of the HP have pentagonal ($N = 11$), square ($N = 8$ and 10) or nearly square (N = 7, 9, and 10) faces. We hypothesize that these non-triangular "defects" in the spherical code solutions are responsible for the increased mobility of these clusters by providing locations where the barrier to rearrangement is low. However, for such small systems, the rearrangements of HPs in a mobile cluster is more appropriately viewed as a rearrangement of the entire cluster rather than a localized rearrangement.

As discussed above, in the second dynamical range, the HPs in an N cluster can almost always be mapped to the points of the N spherical code solution, but that mapping does not remain invariant. We examine the N = 5-12 clusters at the Λ^c at which each cluster is first observed to be measurably mobile per Fig. 10.7 to understand how the HPs in a cluster rearrange. We find (Fig. 10.8) that the N = 5-11 clusters each have a single unique (discounting reflections or rotations) rearrangement, which permutes the HPs over the spherical code points. The $N = 12$ cluster has two unique rearrangements. Short movies of these rearrangements can be found online in the supplemental material. For the $N = 5$ and $N = 11$ clusters, which have structures equivalent to the $N + 1$ spherical code minus a single point, a rearrangement consists of a single HP "hopping" a gap to the available $N + 1$ point. (The reason the $N = 5$ is found in this particular configuration is discussed in the next section.) The clusters N = 6, 10, and 12 exhibit a permutation whereby a ring of HPs rotate relative to the cluster in a manner resembling a twist of a *Rubik's Cube*^{TM}. The clusters N = 7, 8, 9, and 12 exhibit a permutation whereby the cluster "buckles" into a new permutation of the spherical code points. We find that the addition of the rearranging

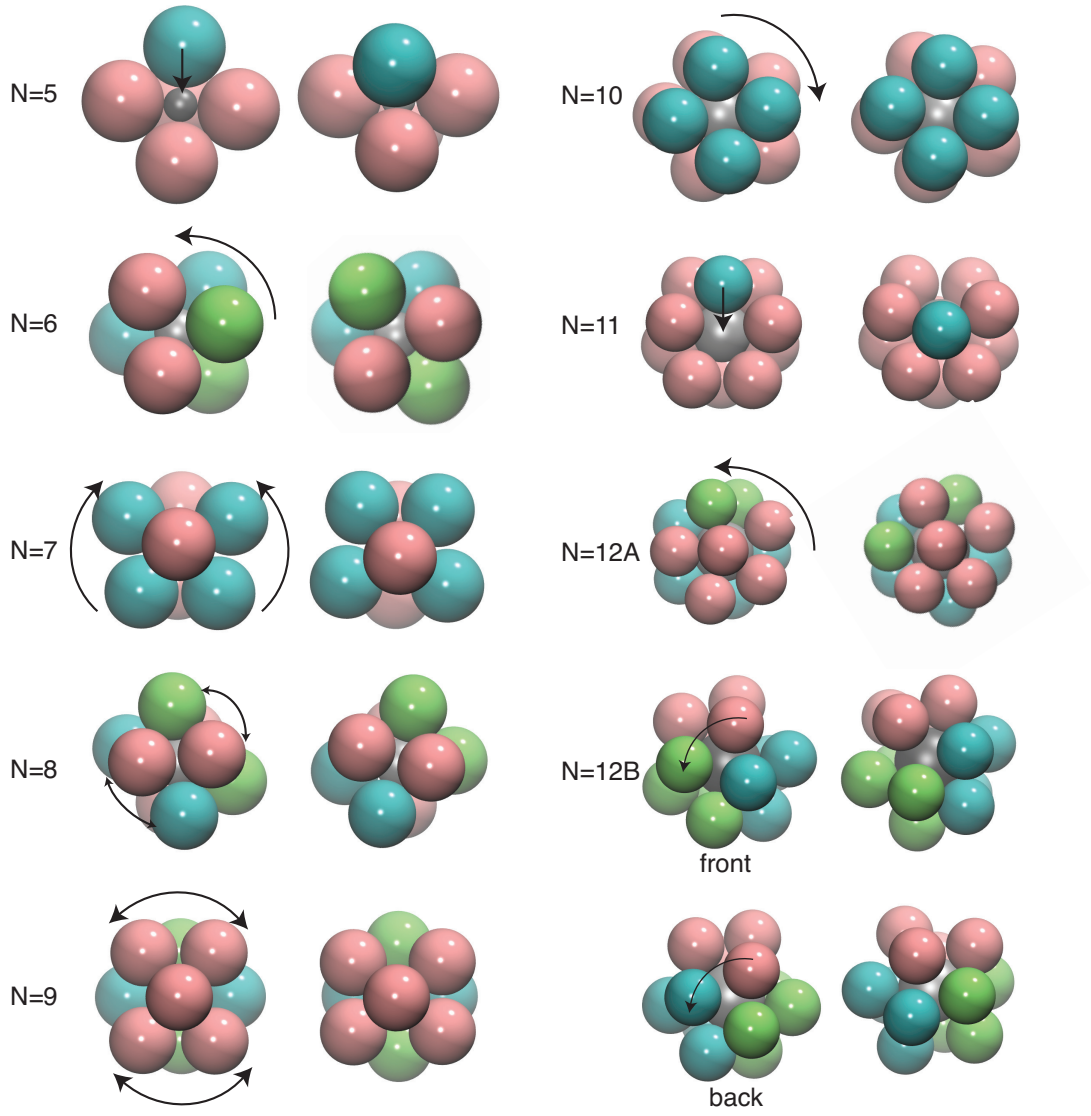


Figure 10.8: The rearrangements of clusters N = 5-12. N = 12 has two rearrangements.

action for N = 5-12 is sufficient to make each cluster ergodic. That is, every possible assignment of each HP to the spherical code points can be explored by the cluster with no inaccessible microstates. This ergodicity is shown, using group theory, in the supplemental materials.

For clusters N = 6, 7, 8, 9, 10, and 12, the rearranging action is a collective motion of particles in the cluster. Although this entropic contribution is not considered by the free energy calculation in Secton 10.2.3, the free energy calculations compare well

with the BD simulations, demonstrating that local rattling is more important than collective modes for some ranges of Λ.

10.3.4 Breaking the degeneracy for $N = 5$

10.3.4.1 BD simulations

The $N = 5$ spherical code has a continuum of solutions ranging from the vertices of a square pyramid to a triangular bipyramid. For dense $N = 5$ clusters at non-zero temperature, we seek the relative likelihood of the cluster adopting particular configurations from the solution continuum. For this, we construct an order parameter that can distinguish between different configurations in our BD simulations.

All $N = 5$ spherical code solutions have two points at opposite poles of the central sphere and differ by the positions of the three remaining points on the equator. The order parameter is constructed by, first, dividing the five HPs into "pole" HPs and "equator" HPs. The neighbor distances, or distance between each HP and the four other HPs is measured. HPs that do not have one neighbor distance that > 1.2 times the distance of the other three neighbor distances are "equator" HPs. Second, the "equator" HP that is closest to other "equator" HPs or has the minimum summed neighbor distances is selected and and its center is labeled A.

Finally, an angle measurement is constructed in the plane of the equator as follows. The centers of the pair of "pole" HPs are labeled P_1 and $P2$. The points A, P_1, and P_2 define a plane S_1. The centers of the two remaining HP are labeled E_1 and E_2. The line through E_1 and E_2 intersects S_1 at E_S and $\hat{n}$ is the normal vector to S_1. A plane S_2 orthogonal to S_1 is constructed from the point E_S, A, and $A+\hat{n}$. The coordinates are translated and rotated so that A and E_S are both on the y-axis of S_2 and A has x-y coordinates $(0, r_0)$, where r_0 is the distance between the center of an HP and the CP. The origin corresponds to the center of the CP. The points E_1 and E_2 are projected to the plane S_2 and the angles ($< \pi/2$) to the x-axis of S_2 is measured. The order

parameter χ is defined as this angle, sampled twice per configuration. Each angle pair uniquely specifies a configuration in the solution continuum. A perfect square pyramid configuration corresponds to two measurements of $\chi = 0$ and a perfect triangular bipyramid configuration corresponds to two measurements of $\chi = \pi/6$ (≈ 0.524) radians. Fig. 10.9a illustrates how the order parameter was constructed, and shows a sampling of the HP positions in the S_2 plane from a simulation at $\Lambda^c = 0.4$. The red circles correspond to the triangular bipyramid positions.

We performed BD simulations of clusters of 5 HP at $\Lambda^c = 0.4142$ and 0.400 with $T^* = 0.02$. Two histograms are shown of the sampled χ at the two ratios, 0.4142 and 0.4 in Fig. 10.9a and 10.9b, respectively. The figures show that the degenerate continuum of $N = 5$ spherical code solutions is broken by the introduction of thermal noise. Surprisingly, we find that the square pyramid is the preferred structure, even over the more symmetrical triangular bipyramid. As the cluster is packed tighter, an even stronger preference for the square pyramid configuration over other configurations emerges.

10.3.4.2 Free Energy

To understand the preference for the square pyramid configuration in the BD simulation we use a free energy calculation, which elucidates the role of entropy in breaking the degeneracy. The more symmetrical triangular bipyramid configuration is used as the reference configuration.

The square pyramid and triangular bipyramid HP clusters are shown in Fig. 10.10(a) and Fig. 10.10(b). In Fig. 10.10(c), using a free energy calculation, the probability of observing the square pyramid relative to the triangular bipyramid is shown at two temperatures as the HP diameter, D_h, approaches the diameter of spheres corresponding to the densest possible packing, $D_{h,N=5}$. For dense clusters, we observe the square pyramid is always the most likely configuration at nonzero temperature. We

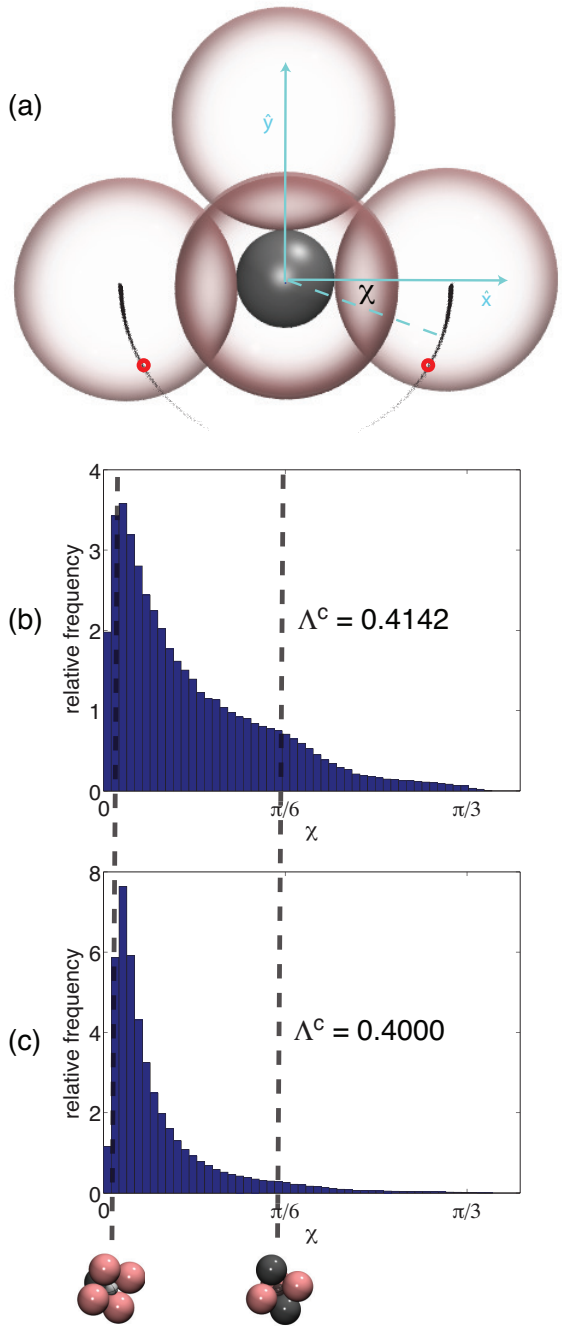


Figure 10.9: (a) The order parameter χ is constructed by measuring the angle of the particles on the equator. Scattered points from a simulation overlay an image of an SP configuration. Red circles indicate the sphere centers of a TBP configuration. In (b) and (c) the distribution of χ sampled in from a BD simulation is shown as a function of the diameter ratio $\Lambda^c = 0.4142$ and 0.4 respectively.

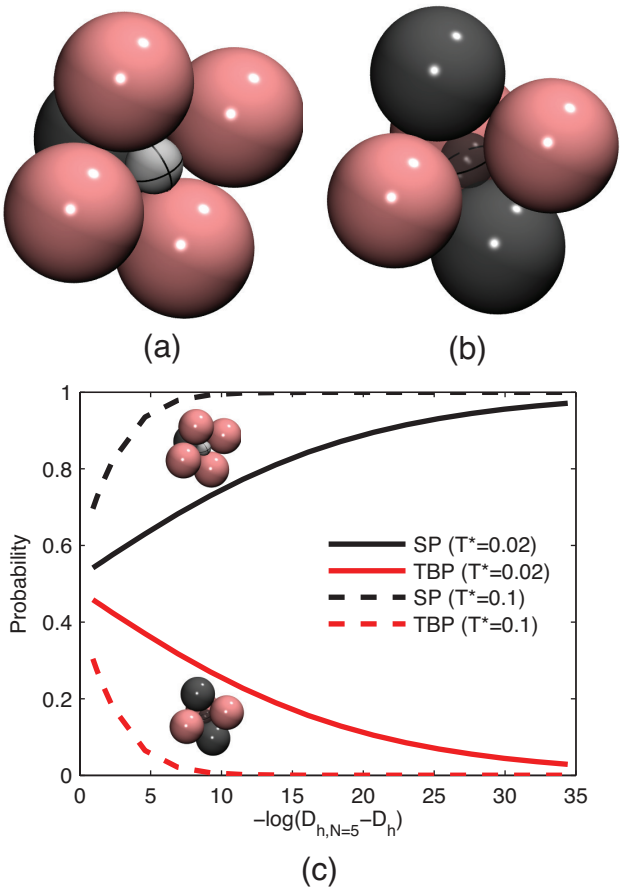


Figure 10.10: (a) The square pyramid (SP) and (b) the triangular bipyramid (TBP) $N = 5$ spherical codes. The jammed and unjammed kissing spheres in each configuration are colored dark grey and pink, respectively. The path that the unjammed spheres can follow is traced on the central sphere. For (b) the central sphere is transparent so the full path around the equator can be seen. In the graph at the bottom, at the low temperature, $T^* = 0.02$, the preference for the SP (black solid) over the TBP (red solid) is evident as the HP diameter approach the limiting packing diameter. This preference (black and red dashed lines) is even stronger at high temperature, $T^* = 0.1$.

find this preference is because the square pyramid has the most vibrational freedom. In Fig. 10.10(a) and Fig. 10.10(b), the locally unjammed HP in a cluster at $\Lambda = \Lambda_{N=5}$ are colored pink and the HP that are locally jammed are colored grey. In the degenerate continuum of $N = 5$ spherical code solutions, only the square pyramid has only one locally jammed HP, and thus, the highest vibrational freedom.

In Fig. 10.7, the $N = 5$ cluster becomes decreasingly mobile as the cluster is

packed tighter, i.e. on decreasing $\Delta\Lambda^c$. Unlike clusters for other values of N, as $\tau \to 0$, the HPs in the $N = 5$ cluster become locally caged for entropic, rather than energetic, reasons.

10.4 Discussion

There are a number of ways the sticky sphere assembly method described above can be extended to create interesting new species of anisotropic particles.

For example, we can now ponder a more general question. Given a desired arrangement of points, what HP-CP interactions and HP-HP interactions will result in self-assembly of the arrangement? The analogous mathematical question was posed by L.L. Whyte in 1952, *"What spherical arrangements [of points] possess extremal properties of any kind?"*[215] Ideally, we seek HP-HP interactions and HP-CP interactions that self-assemble repeatable and desirable patterns of HPs on the CP.

Cohn and Kumar[224] show that all potential energy functions of distance that are completely monotonic, such as inverse power laws, share a subset of universally optimal solution configurations. If the function is strictly completely monotonic, then the universally optimal solution is also unique. For points on the surface of a sphere, the only known universally optimal solutions are[224,225] N = 1-4, 6, and 12; that is, a single point, antipodal points, points forming an equilateral triangle on the equator, and tetrahedral, octahedral, and icosahedral arrangement of points. For our purposes, this means certain desired point arrangements (e.g. a ring of 12 points distributed around the equator of a sphere such as modeled in reference[2]) are likely to be inherently difficult to achieve from HP-HP interactions. Restricting themselves to isotropic pair potentials and identical particles, Cohn and Kumar[226] constructed separate decreasing convex potential energy functions that have cubic (N=8) and dodecahedral (N=20) configurations as their minimum. Thus references[224–226] imply that to assemble certain clusters, it will be necessary to use more complicated

HPs with carefully constructed potentials, including non-completely monotonic or anisotropic interactions.

Using an alternative approach, complex clusters may also be possible by simply adding stages to the assembly process. For example, if, after the terminal N-cluster of Fig. 10.1 is created, the bath of HPs is replaced by a bath of new HPs coated with the same complementary material as the CP, a second shell of spheres can be added to the first. The structure of this shell will also depend on the entropy and energy of the cluster at a given temperature. If the HPs in the second shell preferentially sit in the interstices of the first shell, the polyhedron they form will be the *dual* of the polyhedron of the first shell. This can make new types of point arrangements possible. For example, the dual of the octahedron is the cube. A cubic arrangement of eight points on the surface of a sphere is not found as a minimum among most common spherical surface functions[38]. A second shell of HPs that preferentially assemble the dual of the first shell of HPs may be a physically more viable method of assembling a cubic arrangement of spheres without requiring the elaborately constructed HP-HP interaction potential of Cohn and Kumar[226].

The results presented here may also be used to guide the synthesis of reconfigurable $N = 5$ clusters. As shown in Chapter 10.3.4 above, a small change in the packing fraction of the $N = 5$ cluster introduces a significant change in the structure of the cluster. Thus, changing the effective diameter of the central particle by a modest amount induces a switch between a relatively isotropic disordered cluster and an anisotropic square pyramidal cluster.

10.5 Conclusion

In this paper we have demonstrated that hard and sticky spheres can self-assemble into terminal N-clusters with interesting and, in some cases, unexpected, anisotropies. These clusters have predictable preferred structures that depend on temperature and

sphere diameter ratio. If assembled directly from a bath at low temperature, certain cluster sizes (e.g. $N = 4, 6, 12$) form robustly, while other clusters occur only over small ranges with relatively mobile structures (e.g. N = 7,9,10) and still others cannot be formed at all (e.g. N = 5,11). A "multi-step" process that assembles the clusters from a bath at a higher temperature, removes the bath, and lowers the temperature may enable these hard-to-form clusters to be formed robustly as well. It may even be possible to adjust the effective diameter of the HP or CP as a step in the assembly process. Our free energy calculations and Brownian (molecular) dynamics predictions of cluster structure provide a guide for designing such a process for optimal yield of a desired cluster size with a well-ordered structure. Clusters fabricated in this way may find use as building blocks for subsequent self-assembly, as templates for manufacturing precisely placed circular patches on the surface of a spherical particle, creating nanocolloidal cages, or fabricating reconfigurable particles.

CHAPTER 11

Conclusion and Outlook

11.1 Conclusion

Using computer simulations to study scientific topics via mathematical models is inherently interdisciplinary work. There is arguably no technology that has advanced as fast as the technology of automated computation over the last half-century. As the tools of computational science are far from mature or static, the interplay between the tools and the science are ignored at a scientist's own peril*.

In contrast, the field of mathematics is arguably the most mature discipline. After all, there is no scientific theory of the Ancient Greeks that has gone uncorrected, but every school child is still taught the Pythagorean Theorem. However, mathematics has still only ever answered the questions that somebody has thought to ask. It is not uncommon to pose a calculation, and then consult the literature for the shoulders of the giants to stand upon, only to discover an empty land.

The intention of this dissertation was to advance the field of the study of the self-assembly of anisotropic nanoparticles by integrating advances in both computational sciences and mathematics. We accelerated Molecular Dynamics simulations, an important modeling tool for studying self-assembly, via GPU computing. In Part I, Chapters 3 and 4 we devised, validated, and demonstrated the performance ca-

*Although jumping on every new promised architectural breakthrough is equally perilous, I have been assured.

pability of methods for modeling anistotropic nanoparticles on a GPU, by designing algorithms for implementing rigid body calculations and Brownian dynamics and dissipative particle dynamics simulations in massively parallel computing environments. These algorithms have applications far exceeding the work of this dissertation and are currently openly available in the HOOMD-blue code package.

In Part II, Chapters 5, 6, and 7, we introduced a new method, *filling*, for optimally modeling anistropic particles from a collection of isotropic rigidly connected potentials. In the problem of optimally filling an arbitrary container, we find a novel mathematical problem for the optimal placement of objects in space. In this work we made considerable headway in understanding this problem. We investigated some of the universal mathematical structures that underly the filling solution space. In two-dimensions, both a heuristic and genetic algorithm were designed to find low N optimal solutions for simple polygons. The heuristic algorithm, which exploits the known and conjectured mathematical structure of the solution space, is able to find higher N optimal solutions efficiently. We also derived solutions for filling polygons as $N \to \infty$ from which approximate high N solutions can be constructed. In three dimensions, we investigated the isosymmetric filling solutions of Platonic solids. We also made some progress in understanding the scaling laws and rates of convergence of filling solutions in three and even higher dimensions for some simple geometries. This mathematical foray also has applications that far exceed the work of dissertation, or mere application to modeling anisotropic nanoparticles. This mathematical optimization has applications in any field where optimized removal or deposition of a material is required.

In part III, we studied specific problems of anisotropic nanoparticle self-assembly. In Chapter 8, we studied the question of the impact of natural variation, or polydispersity, on the spontaneous formation of ordered structures in polymer tethered-nanospheres. We found that the packing motifs in the domain of the attractive

nanosphere domain determined the tolerance of the mesoscale structure to polydispersity. In Chapter 9, we introduced a mathematical technique for modeling and then studying the packing density of these domains in these mesoscale structures. And finally, in Chapter 10 we investigated the thermodynamically driven packing of spheres around a central sphere. Using both MD simulations and free energy calculations, we showed that the finite terminal clusters which have structures that can be predicted from a well-known mathematical sequence, are a viable way to self-assemble anisotropic clusters.

We now address several areas of further work, focusing primarily on Part II and III of this dissertation.

11.2 Outlook

11.2.1 Filling in two and three dimensions and beyond

As discussed in chapters 6 and chapters 7 there are still many open mathematical question regarding filling. Here is a brief list.

1. Can conjectures 6.2.1, 6.2.2, and 6.2.3 be proven regarding the properties of optimal filling solutions for polygons or generalized 2D shapes?

2. Can an expression for the unfilled area between two discs along a generalized medial axis path (changing radius function and curvature) be derived?

3. In three dimensions, how do centers optimally pack between two intersecting planes? In an intersection lens? In the sheet structures found in a concave polyhedra? In a generalized medial axis structure of an arbitrary three-dimensional container?

4. Is there a deeper mathematical way to understand why the N=8 solution of a cube has tetrahedral, but not octahedral, symmetry? Are there other exam-

ples of non-trivial optimal solutions that belong to symmetry groups that are subgroups of the symmetry of the shape but not C_1?

5. Can a heuristic be devised that can optimally fill a three-dimensional shape? Do the theorems and conjectures developed in two dimensions have three-dimensional parallels that can also be exploited to find optimal solutions rapidly?

11.2.2 Packing of attractive 3D particles

The research of Chapter 8 was a small exploration in the field of tethered nanoparticles. Research in the Glotzer group has investigated a number of geometrically simple tethered nanoparticles, such as plates, spheres, rods, and even rudimentary cubes. As tethered particles become more complicated, e.g. are tethered to one another and even have different shapes from their tether partners, what mesoscale structures result? The landscape of possibility has the potential to be richer and more diverse than that of block copolymers. This work is currently being carried on in the Glotzer group. The methods of Part II, will be used to model more complicated nanoparticle geometries. However, more efficient ways to automate the search of a vast number of possible dimensions are still waiting to be realized.

11.2.3 Packing of shaped objects around a central sphere

In Chapter 10, hard sphere halo particles which interact isotropically were packed around a sticky sphere central particle. Restricted to such, solutions could be compared to a known mathematical sequence. However if the central or halo particle are no longer spheres, then the packing does not correspond to any mathematical sequence of points that we are aware of. A limited number of computational experiments suggests that even slightly perturbing the spherical structure of the halo particles can have a significant impact on the packing structure, the diversity of structures, and the role of entropy in determining the dominant structure.

APPENDICES

APPENDIX A

Mathematical Derivations

A.1 Distribution function along a medial axis branch with no curvature and a linear radius function

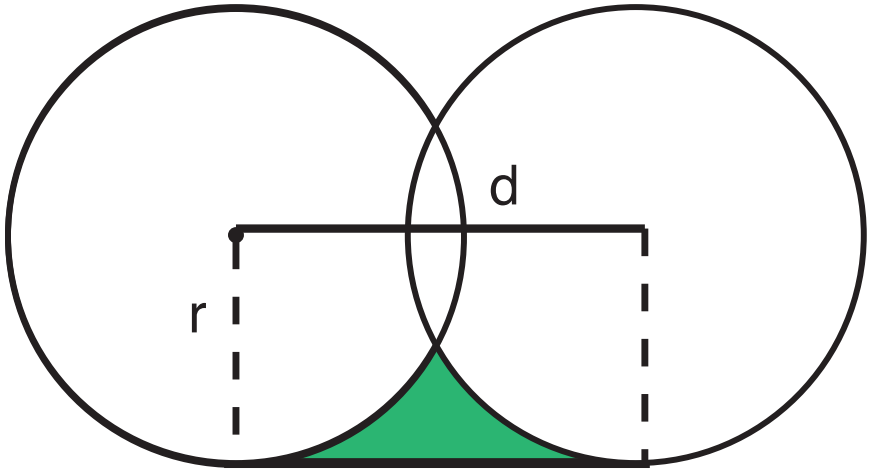


Figure A.1: The area shaded in green is the uncovered area between two discs of the same radius and the polygon edge.

We will calculate the area between two discs along a medial axis branch generated by two polygon edge parents as the two discs approach each other.

Figure A.1 shows two overlapping maximal discs, of the same radius, separated by a distance d, and one of the two lines tangent to both discs. The green region is

the uncovered area in between the discs and the tangent line. As $d \to 0$, what is the area, A of the green region?

$$A = \text{rectangle} - 2 \text{ quarter circles} + \tfrac{1}{2}\text{lens} \tag{A.1}$$

$$= dr - \frac{\pi}{2}r^2 + r^2 cos^{-1}\left(\frac{d}{2r}\right) - \frac{dr}{2}\sqrt{1 - \left(\frac{d}{2r}\right)^2} \tag{A.2}$$

Using a Taylor Expansion of acosine and the square root

$$A = dr - \frac{\pi}{2}r^2 + r^2\left(\frac{\pi}{2} - \frac{d}{2r} - \frac{1}{6}\left(\frac{d}{2r}\right)^3 - \frac{3}{40}\left(\frac{d}{2r}\right)^5\right) \tag{A.3}$$

$$- \frac{dr}{2}\left(1 - \frac{1}{2}\left(\frac{d}{2r}\right)^2 - \frac{1}{8}\left(\frac{d}{2r}\right)^4\right) \tag{A.4}$$

$$A = \frac{d^3}{24r} + O(d^5) \tag{A.5}$$

We now will approximate the uncovered area between two discs and the polygon edges for a radius function is that is not a constant, by bounding the answer between and upper and lower bound. Make one of the discs of Figure A.1 larger by $\Delta r = dr'$, as per the Figure A.2(a). The distance between the two centers is still defined as d. Put a cotangent disc of radius R at the point of tangency of the both discs per Figure A.2(b) and Figure A.2(c). If $R = r$ or if $R = r + dr'$, the centers are now $d\sqrt{1 - r'^2}$ apart. The uncovered area of Figure A.2(a) is bound between Figure A.2(b) and Figure A.2(c) or between $\frac{1}{24}\frac{d^3(\sqrt{1-r'^2})^3}{r}$ and $\frac{1}{24}\frac{d^3(\sqrt{1-r'^2})^3}{r+dr'}$. As $d \to 0$, the area uncovered is

$$A_{uncovered} = \approx \frac{1}{24}\frac{d^3(\sqrt{1 - r'^2})^3}{r}. \tag{A.6}$$

The area is then doubled to account for the identical uncovered piece on the other side due to the other tangent line (i.e. polygon edge).

We now observe that $d = \frac{1}{\rho}$ where ρ is the density of disc centers along the branch.

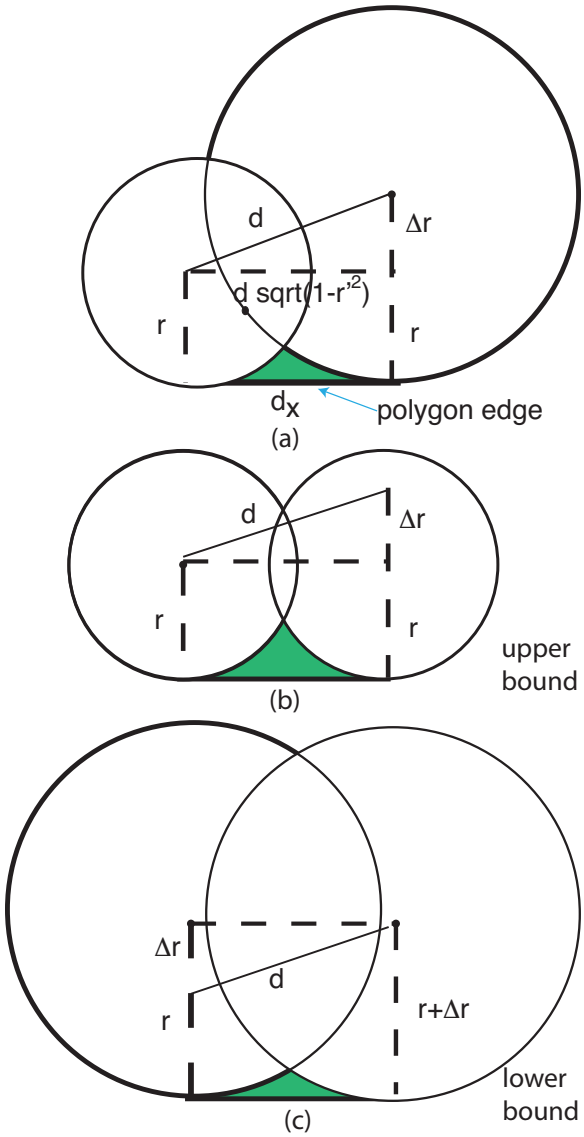


Figure A.2: (a) The area shaded in green is the uncovered area between two discs of different radius and the polygon edge. (b) The area between two small circles and (c) two large circles provide and upper and lower limit for the shaded area.

To determine the total uncovered area along a branch of length T, we would sum all the uncovered areas that are at density ρ along the branch, or

$$A = \int_0^T \frac{(1-r'^2)^{3/2}\,\rho dt}{12r\rho^3} = \int_0^T \frac{(1-r'^2)^{3/2}\,dt}{12r\rho^2} \tag{A.7}$$

A.2 Distributions along the parabolic medial axis branch of a Polygon

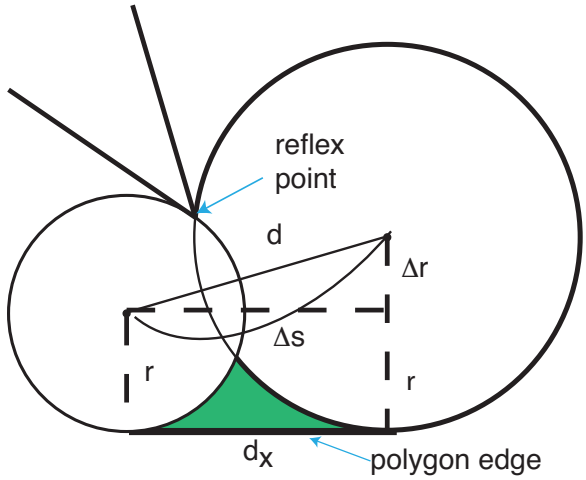


Figure A.3: The area shaded in green is the uncovered area between two discs of the different radius and the polygon edge along a parabolic path.

For concave polygons, the medial axis branch associated with discs tangent to the reflex point and an edge of the polygon is a parabolic curve. The reflex point forms the focus of the parabola and the polygon edge the directrix. If the ends of such a parabolic curve are occupied, all the uncovered area is between the directrix and the set of overlapping discs distributed along the branch. The parabola has both a changing radius function and changing curvature along the branch.

First we note that for two discs that have centers of distance d apart, the uncovered area between the discs and the directerix is the same as Equation A.6.

$$\text{Area} \approx \frac{d^3}{24r}\left(1 - r'^2\right)^{3/2} \tag{A.8}$$

The local density of centers is $\rho = 1/\Delta s$, where Δs is the arc-length between the two centers. However, as $d \to 0$, $d \approx \Delta s$ so,

$$\text{Area} \approx \frac{1}{24\rho^3 r}\left(1 - r'^2\right)^{3/2}. \tag{A.9}$$

Using a parameterization of the parabola, where $y = at^2$, x $= 2at$, $r = at^2 + a$, the arc length $s(t) = a\left(t\sqrt{1+t^2} + \sinh^{-1}t\right)$, and the curvature $\kappa(t) = \frac{1}{2a}\left(1+t^2\right)^{-3/2}$.

then,

$$\frac{dr}{ds} = \frac{dr}{dt}\frac{dt}{ds} = \frac{2at}{2a\sqrt{1+t^2}} = \frac{t}{\sqrt{1+t^2}} \tag{A.10}$$

Note that $r' < 1$, which is a general property of r' of a medial axis.

So

$$(1 - r'^2)^{3/2} = \left(\frac{1}{1+t^2}\right)^{3/2} = 2a\kappa. \tag{A.11}$$

Now Equation A.9 is equal to

$$\text{Area} \approx \frac{1}{24\rho^3}\left(\frac{2r_0\kappa}{r}\right) \tag{A.12}$$

where r_0 is the smallest radius of the parabola, or $r_0 = a$. Or,

$$\text{Area} \approx \frac{1}{24r_0\rho^3}\left(\frac{1}{1+t^2}\right)^{5/2}. \tag{A.13}$$

If the parabola is defined from t_a to t_b, then the total uncovered area is

$$\int_{t_a}^{t_b} \text{Area} \cdot \rho dt = \int_{t_a}^{t_b} \frac{1}{24r_0\rho^2}\left(\frac{1}{1+t^2}\right)^{5/2} dt. \tag{A.14}$$

A.3 Constant curvature, constant radius function

Consider the case of two identical discs are separated by a branch of length d and curvature $\approx \kappa$. What is the uncovered area between them? See Figure A.4. The uncovered area is the the area swept between the two arcs tangent to the discs minus the area covered by the discs within the area swept, or twice the green area of Figure A.4b. Let θ be the angle $\angle ABC$, θ_c be angle $\angle DCE$. Let d_{cl} be the chord length between A and C. Then $\theta = d\kappa$, $d_{cl} = \frac{2}{\kappa}\sin\left(\frac{d\kappa}{2}\right)$, and $\theta_c = 2\cos^{-1}\left(\frac{1}{\kappa r}\sin\left(\frac{d\kappa}{2}\right)\right)$. The

area of the green region of Figure A.4b, is equal to the half disc minus the grey area, or $\frac{\pi r^2}{2} - \frac{r^2}{2}(\theta_c - sin(\theta_c))$.

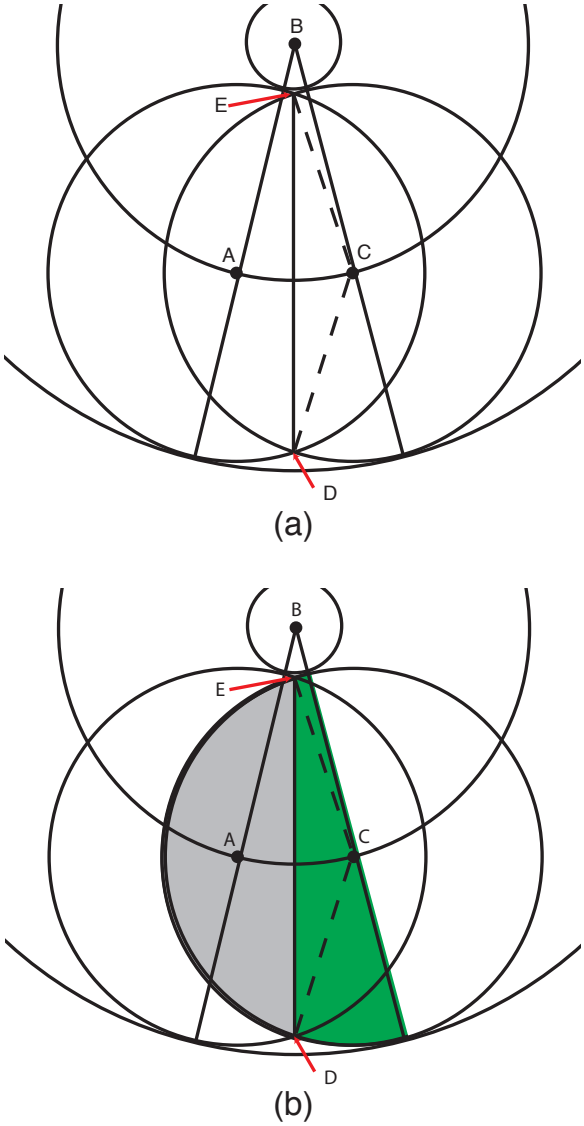


Figure A.4: The uncovered area between two discs on a branch of constant radius function, constant curvature.

$$A_{uncovered} = \left(\left(\frac{1}{\kappa} + r\right)^2 - \left(\frac{1}{\kappa} - r\right)^2\right)\frac{d\kappa}{2} - \pi r^2 + r^2(\theta_c - \sin(\theta_c)) \quad \text{(A.15)}$$

$$= 2rd - \pi r^2 + r^2(\theta_c - \sin(\theta_c)) \quad \text{(A.16)}$$

Expanding the third term

$$r^2\theta_c - r^2\sin(\theta_c) \tag{A.17}$$

$$2r^2\cos^{-1}\left(\tfrac{1}{\kappa r}\sin\left(\tfrac{d\kappa}{2}\right)\right) - r^2\sin\left(2\cos^{-1}\left(\tfrac{1}{\kappa r}\sin\left(\tfrac{d\kappa}{2}\right)\right)\right) \tag{A.18}$$

$$2r^2\cos^{-1}\left(\tfrac{1}{\kappa r}\sin\left(\tfrac{d\kappa}{2}\right)\right) - 2r^2\sin\left(\cos^{-1}\left(\tfrac{1}{\kappa r}\sin\left(\tfrac{d\kappa}{2}\right)\right)\right)\left(\tfrac{1}{\kappa r}\sin\left(\tfrac{d\kappa}{2}\right)\right) \tag{A.19}$$

$$2r^2\left(\cos^{-1}\left(\tfrac{1}{\kappa r}\sin\left(\tfrac{d\kappa}{2}\right)\right) - \sqrt{1-\left(\tfrac{1}{\kappa r}\sin\left(\tfrac{d\kappa}{2}\right)\right)^2}\left(\tfrac{1}{\kappa r}\sin\left(\tfrac{d\kappa}{2}\right)\right)\right) \tag{A.20}$$

Substituting a Taylor series expansion for the inverse cosine and square root

$$2r^2\left(\pi/2 - \left(\tfrac{1}{\kappa r}\sin\left(\tfrac{d\kappa}{2}\right)\right) - \tfrac{1}{6}\left(\tfrac{1}{\kappa r}\sin\left(\tfrac{d\kappa}{2}\right)\right)^3 - \left(1-\tfrac{1}{2}\left(\tfrac{1}{\kappa r}\sin\left(\tfrac{d\kappa}{2}\right)\right)^2\right)\left(\tfrac{1}{\kappa r}\sin\left(\tfrac{d\kappa}{2}\right)\right)\right) \tag{A.21}$$

$$2r^2\left(\pi/2 - \left(\tfrac{2}{\kappa r}\sin\left(\tfrac{d\kappa}{2}\right)\right) + \tfrac{2}{3}\left(\tfrac{1}{\kappa r}\sin\left(\tfrac{d\kappa}{2}\right)\right)^3\right) \tag{A.22}$$

So

$$A_{uncovered} = 2rd - \frac{4r}{\kappa}\sin\left(\frac{d\kappa}{2}\right) + \frac{1}{\kappa^3 r}\frac{4}{3}\left(\sin\left(\frac{d\kappa}{2}\right)\right)^3 \tag{A.23}$$

Substituting a Taylor series expansion for the the sine terms.

$$A_{uncovered} = 2rd - \frac{4r}{\kappa}\left(\left(\frac{d\kappa}{2}\right) - \frac{1}{6}\left(\frac{d\kappa}{2}\right)^3\right) + \frac{1}{\kappa^3 r}\frac{4}{3}\left(\frac{d\kappa}{2}\right)^3 \tag{A.24}$$

$$= \frac{1}{12}d^3\kappa^2 r + \frac{1}{12}\frac{d^3}{r} \tag{A.25}$$

So the uncovered area along the whole length of the branch is ($d = \frac{1}{\rho}$)

$$A = \int_0^T \frac{1}{12}\left(\kappa^2 r + \frac{1}{r}\right)\frac{dt}{\rho^2} = \frac{T^3}{12N^2}\left(\kappa^2 r + \frac{1}{r}\right) \tag{A.26}$$

where N is the total number of discs distributed over the branch section. Using equation (10) and (14) of the main paper, if $M(G)$ can be broken into branch sections

with approximately constant r and curvature, then the fractional distribution of the N discs over each section can be determined such that the fraction f_k discs distributed over a section k of length T_k is

$$f_k \propto T_k \left(\kappa_k^2 r_k + \frac{1}{r_k} \right)^{1/3} \tag{A.27}$$

We observe that, in general, the density of discs is higher in regions of high curvature.

A.4 Deriving distribution of circles

If we divide a $M(G)$ into k branches we can predict what fraction of the discs (N_i/N) will be distributed over each branch i as $N \to \infty$.

$$A = \sum_1^k A_i(N_i) \tag{A.28}$$

$$N = \sum_1^k N_i \tag{A.29}$$

Since we have distributed our discs optimally, we can treat $A_i(N)$ as continuous function and

$$\frac{\partial A_i}{\partial N_i} - \frac{\partial A_j}{\partial N_j} = 0, \forall j \neq i \tag{A.30}$$

Arbitrarily setting $j = k$,

$$\frac{\partial A_i}{\partial N_i} - \frac{\partial A_k}{\partial N_k} = -2\frac{\mathcal{C}_i}{N_i^3} + 2\frac{\mathcal{C}_k}{N_k^3} = 0 \tag{A.31}$$

$$N_i = \left(\frac{\mathcal{C}_i}{\mathcal{C}_k} \right)^{1/3} N_k \tag{A.32}$$

$$f_i = \frac{N_i}{N} = \frac{(\mathcal{C}_i)^{1/3}}{(\mathcal{C}_1)^{1/3} + (\mathcal{C}_2)^{1/3} + ..(\mathcal{C}_k)^{1/3}}. \tag{A.33}$$

A.5 How many ways?

Per conjecture 6.2.1, to find the optimal N filling solution, a maximum must be generated for every way of N discs and K pieces. How many maxima searches is this? Assume that the K pieces have J junctions, $J < K$. For $m \in \mathbb{N}, 0 \leq m \leq J$, there are $\binom{J}{m}$ ways to occupy the junctions, leaving $N - m$ remaining discs to allocate over the $K - J$ remaining pieces. A weak composition is a way of partitioning an integer into a sequence of non-negative integers, where order matters. The number of weak compositions of $N - m$ discs over $K - J$ pieces is $\binom{N-m+K-J-1}{K-J-1}$.

Thus the number of ways to be searched is equal to $\sum_{m=0}^{min(J,N)} \binom{J}{m}\binom{N-m+K-J-1}{K-J-1}$.

$$\sum_{m=0}^{min(J,N)} \binom{J}{m}\binom{N-m+K-J-1}{K-J-1} = \sum_{m=0}^{J} \frac{J!(N-m+K-J-1)!}{m!(J-m)!(K-J-1)!(N-m)!} \tag{A.34}$$

$$\in O\left(N^{K-J-1}\right) \tag{A.35}$$

A.6 Unfilled volume between two balls in a hypercone

To derive an expression for the unfilled volume between two balls of radii r and s in a hypercone, as shown in Figure A.5, we first establish the following expressions and definitions. All expressions below are generalized to n dimensions.

volume of n-sphere of radius r,

$$\frac{\pi^{\frac{n}{2}}}{\frac{n}{2}!} r^n \tag{A.36}$$

volume of (n-1)-sphere of radius r,

$$\frac{\pi^{\frac{n-1}{2}}}{\frac{n-1}{2}!}r^{n-1} \tag{A.37}$$

The volume of a cone is 1/2 height $\cdot$ (n-1)-sphere

Per Figure A.5, A "spherical cone" is the surface of revolution obtained by cutting a conical "wedge" with vertex at the center of a sphere out of the sphere. It is therefore a cone plus a spherical cap.

volume of spherical cone of n-sphere of angle φ of radius r,

$$\frac{\pi^{\frac{n}{2}}}{\frac{n}{2}!}r^n\frac{\int_0^\beta(\sin\varphi)^{n-2}\partial\varphi}{\int_0^\pi(\sin\varphi)^{n-2}\partial\varphi} = \frac{\pi^{\frac{n}{2}}}{\frac{n}{2}!}r^n\frac{\int_0^\beta(\sin\varphi)^{n-2}\partial\varphi}{\frac{\frac{n-3}{2}!}{\frac{n-2}{2}!}\sqrt{\pi}} = \frac{\pi^{\frac{n-1}{2}}}{\frac{n-1}{2}!}\frac{n-1}{n}r^n\int_0^\beta(\sin\varphi)^{n-2}\partial\varphi \tag{A.38}$$

A positive "ice cream cone" is a cone attached to half a sphere, such that the cone and sphere is tangent. Namely, positive ice cream = cone + sphere - spherical cone

A negative "ice cream cone" is a cone minus half a sphere, such that the cone edge and missing half sphere is tangent. Namely, negative ice cream = cone − spherical cone

A lens is the intersection between two spheres. The volume of a lens can be expressed as the sum of two spherical cones minus two flush cones, that is, a revolved triangle.

Let, $\beta = \theta/2$, and observe that $s - r = d\sin\beta$.

In figure A.5, the uncovered volume can now be expressed as

uncovered volume = negative ice cream − positive ice cream + lens

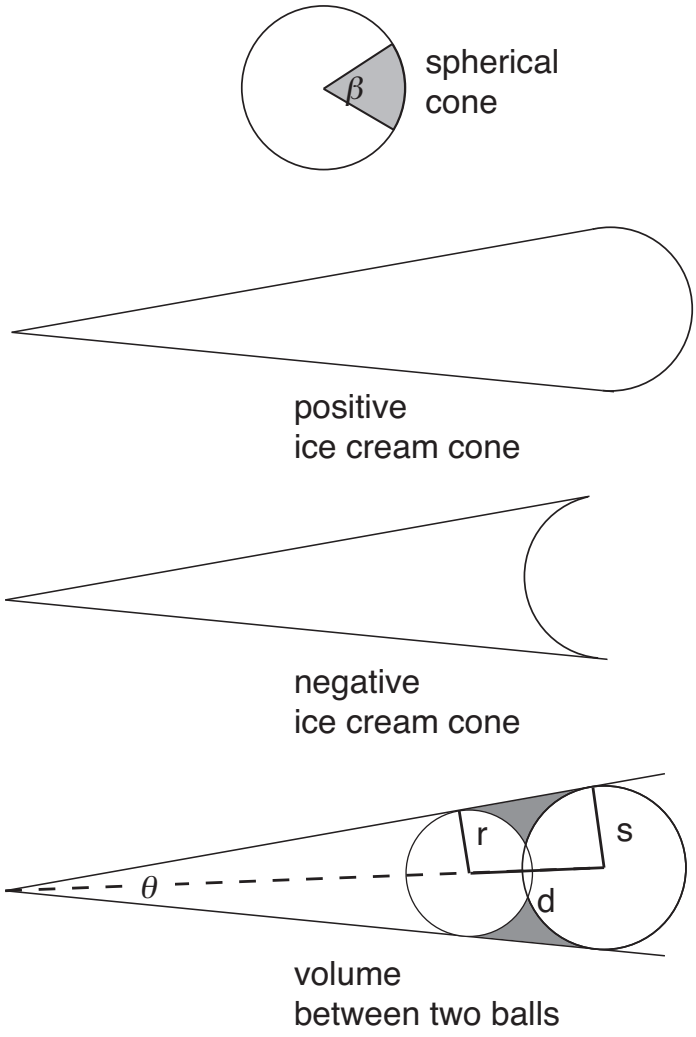


Figure A.5: The geometric n-dimensional pieces of the inclusion exclusion formula include a spherical cone, a positive ice cream cone and a negative ice cream cone. (Bottom) We solve for the shaded region, the volume between two n-balls in a n-dimensional hypercone.

A.6.0.1 negative ice cream cone

Negative Ice Cream Cone = $\frac{1}{n}$ height $\cdot$ sphere[n–1, $s\cos\beta$] - spherical cone[n, s, $\frac{\pi}{2}$–β]

$$\frac{1}{n}(\frac{s}{\sin\beta})(\frac{\pi^{\frac{n-1}{2}}}{\frac{n-1}{2}!}(s\cos\beta)^{n-1}) - \frac{\pi^{\frac{n-1}{2}}}{\frac{n-1}{2}!}\frac{n-1}{n}s^n \int_0^{\frac{\pi}{2}-\beta}(\sin\varphi)^{n-2}\partial\varphi \qquad \text{(A.39)}$$

A.6.0.2 positive ice cream cone

Positive ice cream $= \frac{1}{n}$ height $\cdot$ sphere$[n-1, r\cos\beta]$ $-$ spherical cone$[n, r, \frac{\pi}{2}-\beta]$ $+$ sphere$[n, r]$

$$\frac{1}{n}(\frac{r}{\sin\beta})(\frac{\pi^{\frac{n-1}{2}}}{\frac{n-1}{2}!}(r\cos\beta)^{n-1}) - \frac{\pi^{\frac{n-1}{2}}}{\frac{n-1}{2}!}\frac{n-1}{n}r^n \int_0^{\frac{\pi}{2}-\beta}(\sin\varphi)^{n-2}\partial\varphi + \frac{\pi^{\frac{n}{2}}}{\frac{n}{2}!}r^n \tag{A.40}$$

A.6.0.3 lens

lens $=$ spherical cone$[n, s, \cos^{-1}\frac{d^2+s^2-r^2}{2ds}]$ $+$ spherical cone$[n, r, \cos^{-1}\frac{d^2+r^2-s^2}{2dr}]$ $-$ revolved triangle.

spherical cone$[n, s, \cos^{-1}\frac{d^2+s^2-r^2}{2ds}] = \frac{\pi^{\frac{n-1}{2}}}{\frac{n-1}{2}!}\frac{n-1}{n}s^n\int_0^{\cos^{-1}\frac{d^2+s^2-r^2}{2ds}}(\sin\varphi)^{n-2}\partial\varphi$

spherical cone$[n, r, \cos^{-1}\frac{d^2+r^2-s^2}{2dr}] = \frac{\pi^{\frac{n-1}{2}}}{\frac{n-1}{2}!}\frac{n-1}{n}r^n\int_0^{\cos^{-1}\frac{d^2+r^2-s^2}{2dr}}(\sin\varphi)^{n-2}\partial\varphi$

area of triangle$[d, r, s] = \frac{1}{4}\sqrt{(d+r+s)(-d+r+s)(d-r+s)(d+r-s)}$

$= \frac{1}{4}\sqrt{((r+s)^2-d^2)(d^2-(s-r)^2)}$

$= \frac{1}{4}\sqrt{((r+s)^2-d^2)(d^2-(d\sin\beta)^2)}$

$= \frac{d}{4}(\cos\beta)\sqrt{(r+s)^2-d^2}$

revolved triangle $= \frac{1}{n}$ height $\cdot$ sphere$[n-1, \frac{2}{d}$ area of triangle$[d, r, s]]$

$= \frac{1}{n}(d)(\frac{\pi^{\frac{n-1}{2}}}{\frac{n-1}{2}!}(\frac{1}{2}(\cos\beta)\sqrt{(r+s)^2-d^2})^{n-1})$

A.6.0.4 Uncovered volume

Putting the above expressions together.

uncovered volume

$= \frac{\pi^{\frac{n-1}{2}}}{\frac{n-1}{2}!}\frac{1}{n}\frac{s^n-r^n}{\sin\beta}(\cos\beta)^{n-1} - \frac{\pi^{\frac{n-1}{2}}}{\frac{n-1}{2}!}\frac{n-1}{n}(s^n-r^n)\int_0^{\frac{\pi}{2}-\beta}(\sin\varphi)^{n-2}\partial\varphi - \frac{\pi^{\frac{n}{2}}}{\frac{n}{2}!}r^n$

$+\frac{\pi^{\frac{n-1}{2}}}{\frac{n-1}{2}!}\frac{n-1}{n}s^n\int_0^{\cos^{-1}\frac{d^2+s^2-r^2}{2ds}}(\sin\varphi)^{n-2}\partial\varphi + \frac{\pi^{\frac{n-1}{2}}}{\frac{n-1}{2}!}\frac{n-1}{n}r^n\int_0^{\cos^{-1}\frac{d^2+r^2-s^2}{2dr}}(\sin\varphi)^{n-2}\partial\varphi$

$-\frac{\pi^{\frac{n-1}{2}}}{\frac{n-1}{2}!}\frac{d}{n}\frac{(\cos\beta)^{n-1}}{2^{n-1}}((r+s)^2-d^2)^{\frac{n-1}{2}}$

$= \frac{\pi^{\frac{n-1}{2}}}{\frac{n-1}{2}!}\frac{d}{n}\frac{s^n-r^n}{s-r}(\cos\beta)^{n-1} - \frac{\pi^{\frac{n-1}{2}}}{\frac{n-1}{2}!}\frac{n-1}{n}(s^n-r^n)\int_0^{\frac{\pi}{2}-\beta}(\sin\varphi)^{n-2}\partial\varphi - \frac{\pi^{\frac{n}{2}}}{\frac{n}{2}!}r^n$

$$+\frac{\pi^{\frac{n-1}{2}}}{\frac{n-1}{2}!}\frac{n-1}{n}s^n\int_0^{\cos^{-1}\frac{d^2+s^2-r^2}{2ds}}(\sin\varphi)^{n-2}\partial\varphi+\frac{\pi^{\frac{n-1}{2}}}{\frac{n-1}{2}!}\frac{n-1}{n}r^n\int_0^{\cos^{-1}\frac{d^2+r^2-s^2}{2dr}}(\sin\varphi)^{n-2}\partial\varphi$$
$$-\frac{\pi^{\frac{n-1}{2}}}{\frac{n-1}{2}!}\frac{d}{n}\frac{(\cos\beta)^{n-1}}{2^{n-1}}\left((r+s)^2-d^2\right)^{\frac{n-1}{2}}$$

A.7 Proving the N=5-12 clusters are ergodic when mobile

Consider the N-th spherical code as having N lattice positions 1,2,3,...N and the spheres on the lattice having identities A,B,C,... In a single snapshot of the system, each sphere is assigned to a lattice point (e.g. 1: A, 2: B, 3: C, ...). The possible ways to permute the assignation of the spheres to lattice points form a permutation group. For a finite set of N elements, the symmetric group of the set is the set of all possible arrangements or all permutations of set, which has $N!$ members. Each permutation group is a subgroup of the symmetric group. If the cluster is not mobile, i.e. all HP are locally caged, then the permutation group of the cluster is the rotational symmetry group of the cluster. When the cluster is mobile, there is are additional group actions that permute the cluster. For N=5-12 below, we show that the permutation group generated by the new action found when the cluster is first observed to be mobile and the rotational symmetry group is the symmetric group. We show this by identifying the generators of the permutation group using disjoint cyclic notation, and then use the software of reference[227] to generate the whole group. If the group generated is the same size as the symmetric group, then it is the symmetric group. In each case, we find that the generated group is the symmetric group.

Thus we show that the mobile N=5-12 clusters are ergodic. That is, over time, every possible configuration of spheres is generated. There is no arrangement that cannot be reached by the permuting move alone. For comparison, this property is not true for the corner and edge cubes of a Rubik's cube. There are configurations of a Rubik's cube that can only be generated by illegally disassembling and reassembling the cube.

A.7.1 N=5

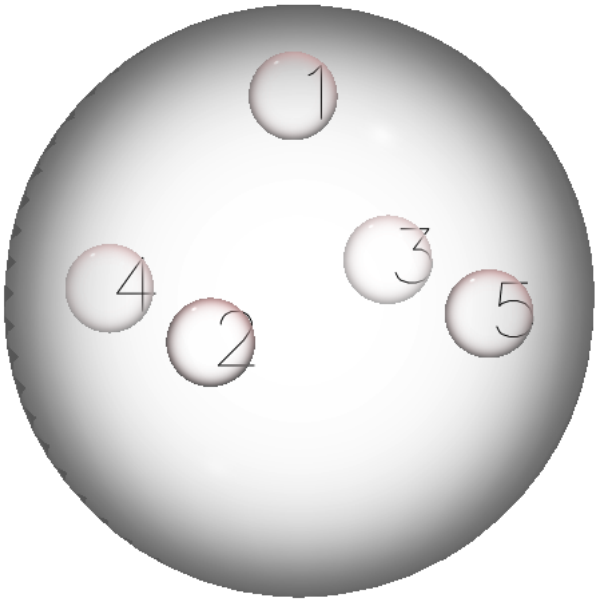


Figure A.6: Labeled spherical code lattice points for N=5 in the square pyramid configuration.

Elements 1,2,3,4,5

The new generator is

- (1,2)(3,1)

Rotational Symmetry Group has generator

- (2,4,3,5)

The resultant permutation group has 120 elements or is equivalent to S_5

A.7.2 N=6

Elements 1,2,3,4,5,6

The new generator is

- (1,2,5)

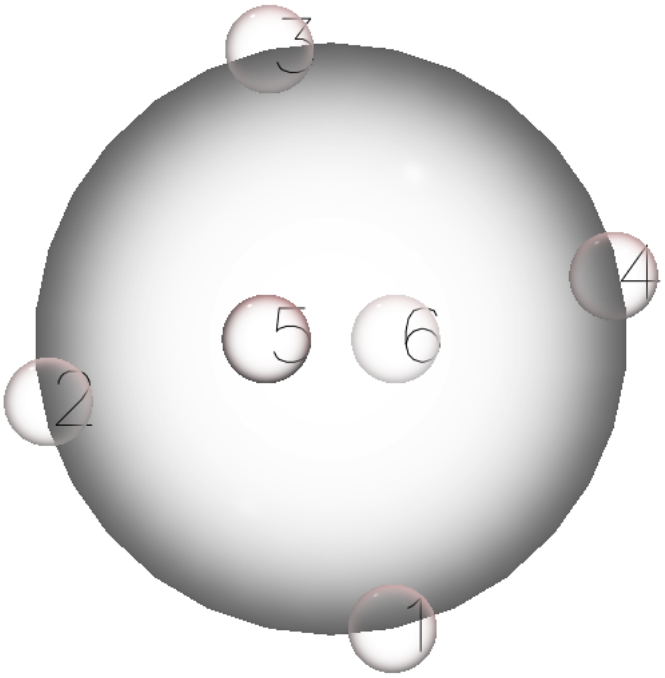


Figure A.7: Labeled spherical code lattice points for N=6.

Rotational Symmetry Group has generator

- (1,5,3,6)
- (2,5,4,6)
- (1,2,3,4)

The resultant permutation group has 720 elements, or is equivalent to S_6.

A.7.3 N=7

Elements 1,2,3,4,5,6,7

The new generator is

- (3,4)(5,6)(7,2)

Rotational Symmetry Group has generator

- (1,4,6)(2,3,5)

The resultant permutation group has 5040 elements, or is equivalent to S_7.

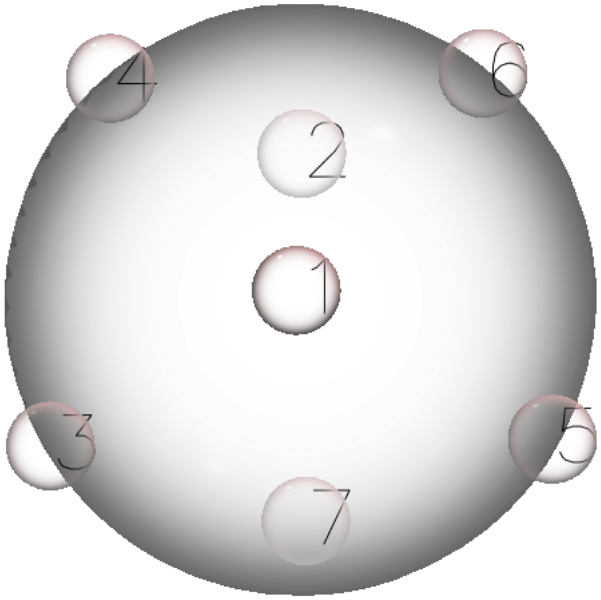


Figure A.8: Labeled spherical code lattice points for N=7.

A.7.4 N=8

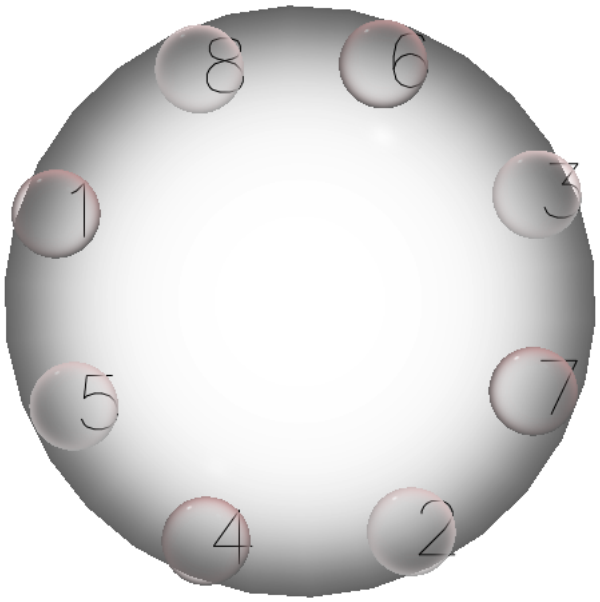


Figure A.9: Labeled spherical code lattice points for N=8.

Elements 1,2,3,4,5,6,7,8

The new generators are

- (5,6,4,8,7)(2,3)
- (4,6,5,7,8)(2,3)

Rotational Symmetry Group has generator

- (1,6,7,4)(8,3,2,5)

The resultant permutation group has 5040 elements, or is equivalent to S_8.

A.7.5 N=9

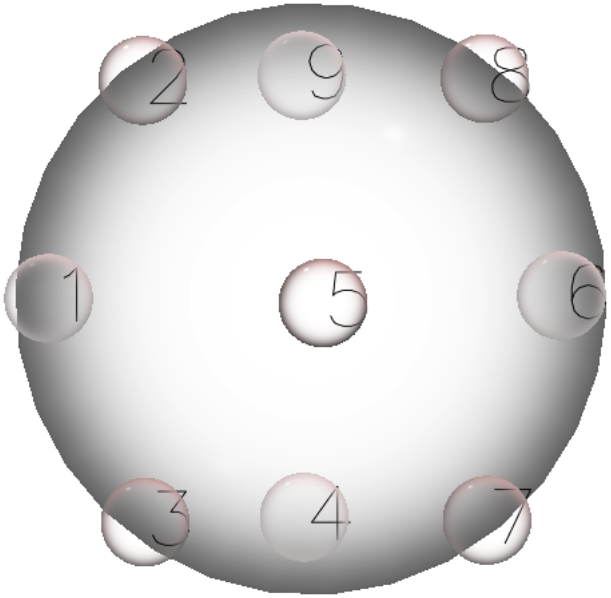


Figure A.10: Labeled spherical code lattice points for N=9.

Elements 1,2,3,4,5,6,7,8,9

The new generator is

- (7,3,2,8)(4,1,9,6)

Rotational Symmetry Group has generators

- (9,4)(3,8)(2,7)
- (9,2,8)(4,3,7)(1,5,6)

The resultant permutation group has 362880 elements, or is equivalent to S_9.

A.7.6 N=10

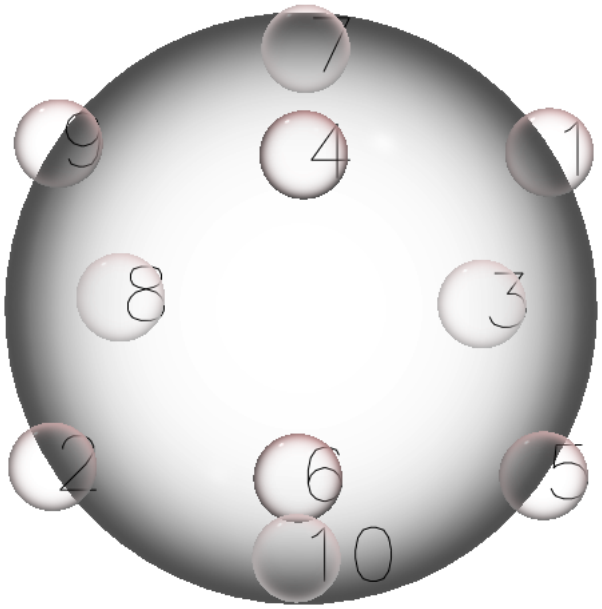


Figure A.11: Labeled spherical code lattice points for N=10.

Elements 1,2,3,4,5,6,7,8,9,10

The new generator is

- (5,4,2)(10,1,9)(7,8,3)

Rotational Symmetry Group has generator

- (4,6)(1,2)(5,9)(7,10)(8,3)

The resultant permutation group has 3628800 elements, or is equivalent to S_{10}.

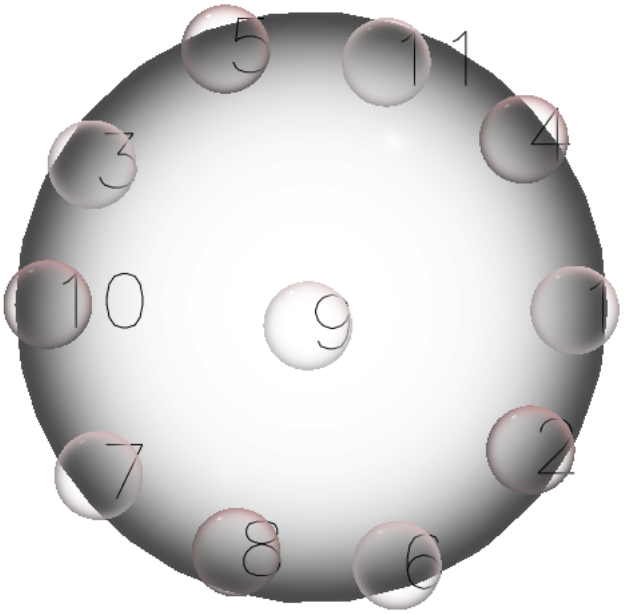


Figure A.12: Labeled spherical code lattice points for N=11.

A.7.7 N=11

Elements 1,2,3,4,5,6,7,8,9,10,11

The new generator is

- (7,9,1,2,8)(11,4,10,3)

Rotational Symmetry Group has generator

- (5,4,2,8,10)(11,1,6,7,3)

The resultant permutation group has 39916800 elements, or is equivalent to S_{11}.

A.7.8 N=12

Elements 1,2,3,4,5,6,7,8,9,10,11,12

The new generators are

- (11,4,2,12,6)

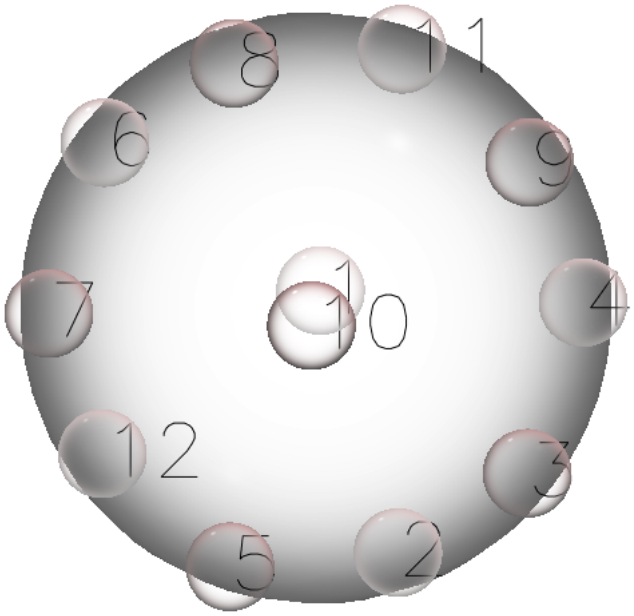


Figure A.13: Labeled spherical code lattice points for N=12

- (4,3,5,12,6,11)

Rotational Symmetry Group has generators

- (11,4,2,12,6)(9,3,5,7,8)
- (11,4,3,10,8)(1,2,5,7,6)
- (9,4,2,5,10)(11,1,12,7,8)
- (8,6,12,5,10)(11,1,2,3,9)
- (9,8,6,1,4)(10,7,12,2,3)

Note that first new generator (the Rubik's Cube twist) and the generators of the rotational symmetry group are not sufficient to generate the symmetric group. However, the second generator and the rotational symmetry group is sufficient to generate the symmetric group. The resultant permutation group has 479001600 elements, or is equivalent to S_{12}.

APPENDIX B

Codes

B.1 Chapter 2

Code for performing GPU-accelerated rigid body simulations and FIRE energy minimizations can be found in the open-source code package HOOMD-blue version 0.10.0[73] and later.

B.2 Chapter 3

Code for performing GPU-accelerated Brownian dynamics and dissipative particle simulations can be found in the open-source code package HOOMD-blue version 0.9.1[73] and later.

B.3 Chapter 4,5,6

All code for performing the genetic algorithm on simple polygons in 2D is contained in the codeblue.umich.edu repository http://codeblue.umich.edu/index/projects/gap-ga in gap-ga/2D.

All code for performing the genetic algorithm and simulated annealing simulations on convex polygons in 3D is contained in the codeblue.umich.edu repository http://codeblue.umich.edu/index/projects/gap-ga in gap-ga/3D.

All code for performing the heuristic algorithm on simple polygons in 2D is contained in the codeblue.umich.edu repository http://codeblue.umich.edu/index/projects/gap-ha

B.4 Chapter 7

The following python script can be run with HOOMD-blue and the "particle_grower" plugin.

B.4.1 Polydisperse tethered nanosphere script

```
from hoomd_script import *
from hoomd_plugins import particle_grower
import math

#Generate a startup file from scratch

# parameters
phi_P = 0.4
n_particle = 2000
myT = 0.3
myPD = 10.0

# One Head
head_num = 1
head_diam = 1.94
```

```
# 8 Tail
tail_num = 8

tns = dict(bond_len=1.0, type=['Head']*head_num + ['Tail']*tail_num, bond="linear",
count=n_particle)

# perform some simple math to find the length of the box
N = len(tns['type']) * tns['count']
Nhead = head_num*tns['count']
Ntail = tail_num*tns['count']
L = math.pow(math.pi * (Nhead * (head_diam*head_diam*head_diam) + Ntail) /
(6.0 * phi_P), 1.0/3.0)

# generate the polymer system
init.create_random_polymers(box=hoomd.BoxDim(L), polymers=[tns],
separation=dict(Head=0.35, Tail=0.35), seed=12345)

# pair potential setup
eps=1.0

# bonds
myfene = bond.fene()
myfene.set_coeff('polymer', k=30.0, r0=1.5, sigma=1.0, epsilon=eps)

# pair interaction between head and tail groups (initially made WCA)
sljWCA = pair.slj(r_cut=2**(1.0/6.0))
```

```
sljWCA.pair_coeff.set('Head', 'Head', epsilon=eps, sigma=1.0)
sljWCA.pair_coeff.set('Head', 'Tail', epsilon=eps, sigma=1.0)
sljWCA.pair_coeff.set('Tail', 'Tail', epsilon=eps, sigma=1.0)
sljWCA.set_params(mode="shift")

# dump every few steps
mol2 = dump.mol2()
mol2.write(filename="TNS.mol2")
dump.dcd(filename="TNS.dcd", period=1000000)

# integrate NVT for a bunch of time steps
bd=integrate.bdnvt(dt=0.01, T=myT, use_diam=True, seed=54321)

# start scaling particle size
grow_em=update.diameter_scaler(r_cut=2**(1.0/6.0))
heads=group.type('Head')
grow_em.add_group(group=heads,growtime=2000, finalD=head_diam)

run(4000)
#Polydisperse the head groups
pd_em=particle_grower.update.pdgrower(r_cut=pow(2.0,1.0/6.0))
pd_em.add_group(group=heads, minD=1.0, PD=myPD, seed=12345)
pd_em.add_move(group=heads, growtime=2000, startf=0.0, stopf=1.0)
pdlog=analyze.log(filename='polydispersity.log', quantities=['polydispersity', 'aver-
age_diameter'], period=100000)
```

```
logger = analyze.log(filename='log_cooldown.pe', period=1000, quantities=['pair_lj_energy'])
outputxml=dump.xml(filename="TNSinit", period=1000000)
outputxml.set_params(position=True, velocity=True, mass=True, diameter=True,
type=True, bond=True)

run(1000000)

#Turn Off WCA head interaction, PD Log and PD'er
sljWCA.disable()
pdlog.disable()
pd_em.disable()

# pair interaction between head and tail groups (heads now attract)
slj = pair.slj(r_cut=2.5))
slj.pair_coeff.set('Head', 'Head', epsilon=eps, sigma=1.0)
slj.pair_coeff.set('Head', 'Tail', epsilon=eps, sigma=1.0, r_cut=2**(1.0/6.0)
slj.pair_coeff.set('Tail', 'Tail', epsilon=eps, sigma=1.0, r_cut=2**(1.0/6.0)
slj.set_params(mode="shift")

run(100000000)
```

B.5 Chapter 8

The radical tessellation code has been incorporated into vor++ software library maintained by Dr. Chris Rycroft and can be found at , http://math.lbl.gov/voro++/ .

B.6 Chapter 9

The following python scripts can be run with HOOMD-blue and the "particle_grower" and "spherical_code_pairs" plugin and using BBB

B.6.1 Spherical code self-assembly code

```
#! /usr/bin/env hoomd


from hoomd_script import *
from hoomd_plugins import spherical_code_pairs
import numpy
import math
from bbb import basic_blocks, packmol, lattice
import copy
import os


# Where the transition occurred using the nominal Dc/Dh
transition = [ [0.1, 3], [0.166,4], [0.364, 6], [0.592, 7], [0.649, 8], [0.745, 9], [0.853, 10], [0.943, 12]]
    # get the job index from PBS_ARRAYID, or return 0 if it is not specified (for test jobs)
def get_array_id():
    pbs_arrayid = os.getenv('PBS_ARRAYID');
        if pbs_arrayid is None:
        return 0
    else:
        return int(pbs_arrayid)-1;
```

```
# setup derived parameters
id = get_array_id();

# minimal ratio of HP to CP ratio = 4;

#Number of Central Particles
N_A= 1000

# ratio of CP Diameter to HP Diameter
CPoverHP = id*(1-0.1)/300 + 0.1;

# Determine what the multiplier is
multiplier = 1;
for val in transition:
    multiplier = val[1];
    if val[0] >= CPoverHP:
        break

#Number of Particles
N_B= multiplier*ratio*N_A

# Halo Sphere
Halo_Diameter = 3.0;

# Central Sphere
Central_Diameter = CPoverHP*Halo_Diameter;
```

```
seed = 98749874

# setup the output directory
out_dir = 'R'+ str(CPoverHP).ljust(10,'0') + '_D' + str(Central_Diameter) + 'd' +
str(Halo_Diameter)

if os.path.exists(out_dir):
    if not os.path.isdir(out_dir):
        raise RuntimeError(out_dir + ' exists and is not a directory')
    else:
        os.mkdir(out_dir);

# Box Length
# Write formula for 25% dense box
phi = 0.25
box_vol = (N_A*(1.0/6.0)*math.pi*math.pow(Central_Diameter,3.0)
+N_B*(1.0/6.0)*math.pi*math.pow(Halo_Diameter,3.0))/phi
box_length = box_vol**(1.0/3.0)

# Make the Sphere A particles
# Making a single lattice of particles that will be fixed
sphereA = basic_blocks.Rod(N=N_A, r0 = 1.0, diameter=Central_Diameter,type='A');

sphereB = basic_blocks.Sphere(diameter=Halo_Diameter,type='B');

# Make the Simulation Box
box = packmol.ConstraintSimulationBoxHOOMD(box_length, box_length, box_length,
```

```
tolerance = max([Halo_Diameter, Central_Diameter]))

# Setup the generator and pick a seed
g = packmol.GeneratorXML(simulation_box=box, seed=1, tolerance = max([Halo_Diameter,
Central_Diameter]), unique_suffix=str(id));

# Put Spheres in the box subject to Constraints
g.addBuildingBlock(sphereB, N_B)

l = int(math.ceil(N_A ** (1.0/3.0)))
num_A=0
inset=box_length/10.0;

for i in xrange(0,l):
    for j in xrange(0,l):
        for k in xrange(0,l):
            if N_A > num_A:
                sphereA.getParticle(num_A).position = (i*box_length/l-box_length/2.0+inset,
j*box_length/l-box_length/2.0+inset, k*box_length/l-box_length/2.0+inset)
            num_A = num_A+1

g.addBuildingBlock(sphereA, 1, constraints=[packmol.ConstraintFixed(0,0,0)])

# Step 3.4: generate the xml file (will overwrite if it already exists!)
g.writeOutput(open(out_dir+'/Init.xml', 'w'));

system = init.read_xml(filename=out_dir+"/Init.xml")
```

```
#Pairwise Interaction Force

lj = pair.slj(r_cut = 2**(1.0/6.0), d_max = max(Central_Diameter,Halo_Diameter))
lj.pair_coeff.set('A', 'A', epsilon=0.0, sigma=1.0)
lj.pair_coeff.set('B', 'B', epsilon=1.0, sigma=1.0)
lj.pair_coeff.set('A', 'B', epsilon=0.0, sigma=1.0)

#Pairwise Interaction Force morse = spherical_code_pairs.pair.shiftedmorse(r_cut=2.5,
d_max = max(Central_Diameter,Halo_Diameter))
morse.pair_coeff.set('A', 'B', D0=5.0, alpha=5.0, dr0 = 0)
morse.pair_coeff.set('A', 'A', D0=0.0, alpha=5.0, dr0 = 0)
morse.pair_coeff.set('B', 'B', D0=0.0, alpha=5.0, dr0 = 0)

# neighbor list exclusions
#nlist.reset_exclusions(exclusions = ['bond', 'diameter'])

anchor = group.type(type='A')
halo = group.type( type='B')
groupAB = group.union(name="ab-particles", a=anchor, b=halo)

integrate.mode_standard(dt=0.005)

bd=integrate.bdnvt(group=halo, T=1.0, seed=seed)
bd.set_params(T = variant.linear_interp(points = [(0, 1.0), (2e5, 0.1)]))

dump.dcd(filename=out_dir+"/trajectory.dcd", period=int(1e5), overwrite=True)
```

```
#imd = analyze.imd(port=12345, period = 100)

xml=dump.xml()
xml.set_params(all=True)

dump.bin(file1=out_dir + '/restart.1.bin.gz', file2=out_dir + '/restart.2.bin.gz', period=1e4)

analyze.log(filename=out_dir+'/mylog.log', quantities=['temperature', 'kinetic_energy', 'potential_energy'], period=100, header_prefix='#', overwrite =True)

# warm up run
run(int(20e6))

xml.write(filename=out_dir+'/Final.xml')

# Last Dump
bin = dump.bin()
bin.write(filename=out_dir + '/continue.bin.gz')
```

B.6.2 HP constrained to a CP surface

```
    from hoomd_script import *
from hoomd_plugins import particle_grower
from hoomd_plugins import spherical_code_pairs
import numpy
import math
```

```
from bbb import basic_blocks, packmol, lattice
import copy
import pickle
import sys

# One Sphere in the Middle of type A

# N spheres attached to it
N_B =int(sys.argv[3])

# Ratio CPoverHP = float(sys.argv[2])
# convert
CPoverHP = (CPoverHP*3.0786 +(3.0786-3.0))/3.0;

T=0.1;

grow_time = int(1e5)
sample_time = int(1e4)
num_samples = 10000

# Halo Sphere Halo_Initial_Diameter = 1.0;
Halo_Diameter = 3.0;

# Central Sphere
Central_Diameter = CPoverHP*(Halo_Diameter)
Radial_Offset = (Central_Diameter + Halo_Diameter)/2.0;
```

```
sphere = basic_blocks.Sphere(diameter = Halo_Initial_Diameter, type = 'B' );
sphere_constraint1 = packmol.ConstraintSphere('inside', 0,0,0, Radial_Offset + 0.1,
applyto = [0])
sphere_constraint2 = packmol.ConstraintSphere('outside', 0,0,0, Radial_Offset, ap-
plyto = [0])

box = packmol.ConstraintSimulationBoxHOOMD(140, 140, 140,
tolerance = Halo_Initial_Diameter)
g = packmol.GeneratorXML(simulation_box=box, seed=1,
tolerance = Halo_Initial_Diameter);

# Put the Central Sphere in the box
g.addBuildingBlock(sphere, N_B, constraints = [sphere_constraint1, sphere_constraint2])
g.addBuildingBlock(basic_blocks.Sphere(type='A', diameter = Central_Diameter),1,
constraints = [packmol.ConstraintFixed(0,0,0)])
g.writeOutput(open('Init.xml', 'w'));

#Start Simulation
system = init.read_xml("Init.xml")

#Pairwise Interaction Force
lj = pair.slj(r_cut = 2**(1.0/6.0),d_max =Halo_Diameter)
lj.pair_coeff.set('A', 'A', epsilon=0.0, sigma=1.0)
lj.pair_coeff.set('B', 'B', epsilon=0.1/T, sigma=1.0)
lj.pair_coeff.set('A', 'B', epsilon=0.0, sigma=1.0)

anchor = group.type(type='A')
```

```
halo = group.type( type='B')
groupAB = group.union(name="ab-particles", a=anchor, b=halo)

#Constrain Sphere
constrain.sphere(group=halo, P=(0,0,0), r=Radial_Offset)

integrate.mode_standard(dt=0.005)
bd=integrate.bdnvt(group=halo, T=T, seed=1234)

# Grower!
run(int(1e4))

grower = particle_grower.update.diameter_scaler(period = 1)
grower.add_group(group = halo, finalD = Halo_Diameter, growtime = grow_time)
run(grow_time)

grower.disable()
del grower

#Dump an example of what the system looks like at the end of the grow.
xml = dump.xml()
xml.set_params(all=True)
xml.set_params(image=False)
xml.write(filename='Small.xml')

#Save Data
filename = 'trajectory_' + str(Central_Diameter/(Halo_Diameter)) + '_0.dcd'
```

```
datadcd = dump.dcd(filename=filename, period=sample_time, overwrite = True)


run(sample_time*num_samples)
```

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Trung Dac Nguyen, Carolyn L. Phillips, Joshua A. Anderson, and Sharon C. Glotzer. Rigid body constraints realized in massively-parallel molecular dynamics on graphics processing units. *Computer Physics Communications*, 182(11): 2307 – 2313, 2011.

[2] Zhenli Zhang and Sharon C. Glotzer. Self-assembly of patchy particles. *Nano Letters*, 4(8):1407–1413, 2004.

[3] Carolyn L. Phillips, Joshua A. Anderson, and Sharon C. Glotzer. Pseudo-random number generation for brownian dynamics and dissipative particle dynamics simulations on gpu devices. *Journal of Computational Physics*, 230(19): 7191 – 7201, 2011.

[4] R. D. Groot, T. J. Madden, and D. J. Tildesley. On the role of hydrodynamic interactions in block copolymer microphase separation. *J. Chem. Phys*, 110: 9739–9749, May 1999. This paper was shown incorrect by[5].

[5] MA Horsch, ZL Zhang, and SC Glotzer. Self-assembly of polymer-tethered nanorods. *Physical Review Letters*, 95(5), 2005.

[6] Carolyn L. Phillips, Christopher R. Iacovella, and Sharon C. Glotzer. Stability of the double gyroid phase to nanoparticle polydispersity in polymer-tethered nanosphere systems. *Soft Matter*, 6(8):1693–1703, 2010.

[7] P K Doye, Jonathan and Lars Meyer. Mapping the magic numbers in binary lennard-jones clusters. *Phys Rev Lett*, 95(6):063401, 2005.

[8] Cyril Stanley Smith. *A History of Metallography: The Development of Ideas on the Structure of Metals before 1890*. MIT Press, 1988.

[9] "There's Plenty of Room at the Bottom" by Richard P. Feynman, lecture at APS meeting, December 29, 1959.

[10] Sharon C. Glotzer and Michael J. Solomon. Anisotropy of building blocks and their assembly into complex structures. *Nature Materials*, 6(7):557–562, August 2007.

[11] Krste Asanovic, Ras Bodik, Bryan C. Catanzaro, Joseph J. Gebis, Parry Husbands, Kurt Keutzer, David A. Patterson, William L. Plishker, John Shalf,

Samuel W. Williams, and Katherine A. Yelick. The landscape of parallel computing research: a view from Berkeley. Technical Report UCB/EECS-2006-183, Electrical Engineering and Computer Sciences, University of California at Berkeley, December 2006.

[12] A. Rahman. Correlations in the motion of atoms in liquid argon. *Phys. Rev.*, 136:A405–A411, Oct 1964.

[13] V.A Shneidman and D.R Uhlmann. Does a Lennard-Jones glass exist? *Journal of Non-Crystalline Solids*, 224(1):86 – 88, 1998.

[14] W. Kob and H. C. Andersen. Testing mode-coupling theory for a supercooled binary Lennard-Jones mixture i: The van Hove correlation function. *Phys. Rev. E*, 51(5):4626–4641, May 1995.

[15] G. H. Ristow. Simulating Granular Flow with Molecular Dynamics. *J. Phys. I*, 2(6):649, 1992.

[16] Go Watanabe, Jun-Ichi Saito, Nobuyuki Kato, and Yuka Tabe. Orientational correlations in two-dimensional liquid crystals studied by molecular dynamics simulation. 134(5):054513, 2011.

[17] Michael P. Allen, Mark A. Warren, Mark R. Wilson, Alain Sauron, and William Smith. Molecular dynamics calculation of elastic constants in gay berne nematic liquid crystals. *J. Chem. Phys.*, 105(7):2850–2858, 1996.

[18] Kurt Kremer and Gary S. Grest. Dynamics of entangled linear polymer melts: A molecular dynamics simulation. *J. Chem. Phys.*, 92(8):5057–5086, 1990.

[19] E. S. Boek, P. V. Coveney, H. N. W. Lekkerkerker, and P. van der Schoot. Simulating the rheology of dense colloidal suspensions using dissipative particle dynamics. *Phys. Rev. E*, 55:3124–3133, Mar 1997.

[20] Makoto Fushiki. Molecular dynamics simulations for charged colloidal dispersions. *J. Chem. Phys.*, 97(9):6700–6713, 1992.

[21] Kristen A. Fichthorn and Yong Qin. Molecular-dynamics simulation of colloidal nanoparticle forces. *Industrial & Engineering Chemistry Research*, 45(16):5477–5481, 2006.

[22] M. R. D'Orsogna, Y. L. Chuang, A. L. Bertozzi, and L. S. Chayes. Self-Propelled Particles with Soft-Core Interactions: Patterns, Stability, and Collapse. *Physical Review Letters*, 96(10):104302+, March 2006. doi: 10.1103/PhysRevLett.96.104302.

[23] M. Shimizu, A. Ishiguro, T. Kawakatsu, Y. Masubuchi, and M. Doi. Coherent swarming from local interaction by exploiting molecular dynamics and stokesian dynamics methods. In *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 2, pages 1614 – 1619 vol.2, oct. 2003.

[24] CR Iacovella, MA Horsch, Z Zhang, and SC Glotzer. Phase diagrams of self-assembled mono-tethered nanospheres from molecular simulation and comparison to surfactants. *Langmuir*, 21(21):9488–9494, 2005.

[25] CR Iacovella, AS Keys, MA Horsch, and SC Glotzer. Icosahedral packing of polymer-tethered nanospheres and stabilization of the gyroid phase. *Phys Rev E*, 75(4), 2007.

[26] Christopher R. Iacovella and Sharon C. Glotzer. Phase behavior of ditethered nanospheres. *Soft Matter*, 5:4492–4498, 2009.

[27] Trung Dac Nguyen, Zhenli Zhang, and Sharon C. Glotzer. Molecular simulation study of self-assembly of tethered v-shaped nanoparticles. *J. Chem. Phys.*, 129 (24):244903, 2008.

[28] Trung Dac Nguyen and Sharon C. Glotzer. Switchable helical structures formed by the hierarchical self-assembly of laterally tethered nanorods. *Small*, 5(18): 2092–2098, 2009.

[29] SC Glotzer, MA Horsch, CR Iacovella, Z Zhang, ER Chan, and X Zhang. Self-assembly of anisotropic tethered nanoparticle shape amphiphiles. *Current Opinion in Colloid and Interface Science*, 2005.

[30] Trung Dac Nguyen, Eric Jankowski, and Sharon C. Glotzer. Self-assembly and reconfigurability of shape-shifting particles. *ACS Nano*, In Press, 2011.

[31] http://lammps.sandia.gov.

[32] S Sastry, DS Corti, PG Debenedetti, and FH Stillinger. Statistical geometry of particle packings.1. algorithm for exact determination of connectivity, volume, and surface areas of void space in monodisperse and polydisperse sphere packings. *Phys Rev E*, 56(5):5524–5532, 1997.

[33] A Okabe. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley, New York, 2000.

[34] F. Aurenhammer. Power Diagrams: Properties, Algorithms and Applications. *SIAM Journal on Computing*, 16(1):78–96, 1987.

[35] M. Gavrilova and J. Rokne. An efficient algorithm for construction of the power diagram from the voronoi diagram in the plane. *International journal of computer mathematics*, 61(1-2):49–61, 1996.

[36] Hiroshi Imai, Masao Iri, and Kazuo Murota. Voronoi diagram in the laguerre geometry and its applications. *SIAM J. Comput.*, 14(1):93–105, 1985.

[37] CH Rycroft, GS Grest, JW Landry, and MZ Bazant. Analysis of granular flow in a pebble-bed nuclear reactor. *Physical Review E*, 74, 2006.

[38] N. J. A. Sloane, with the collaboration of R. H. Hardin, W. D. Smith and others, Tables of Spherical Codes, Coverings, Maximum Volume of Convex Hulls, and Minimal Energy Arrangements published electronically at www.research.att.com/ njas/.

[39] D. Frenkel and B. Smit. *Understanding Molecular Simulations*. Academic Press, 2nd edition, 2002.

[40] Mirjam E. Leunissen and Daan Frenkel. Numerical study of dna-functionalized microparticles and nanoparticles: Explicit pair potentials and their implications for phase behavior. *The Journal of Chemical Physics*, 134(8):084702, 2011.

[41] Allen, M. P. and Tildesley, D. J. *Computer Simulation of Liquids*. Oxford science publications. Oxford University Press, USA, June 1989.

[42] John D. Weeks, David Chandler, and Hans C. Andersen. Role of repulsive forces in determining the equilibrium structure of simple liquids. *The Journal of Chemical Physics*, 54(12):5237–5247, 1971.

[43] Philip M. Morse. Diatomic molecules according to the wave mechanics. ii. vibrational levels. *Phys. Rev.*, 34:57–64, Jul 1929.

[44] GS Grest and K Kremer. Molecular-dynamics simulation for polymers in the presence of a heat bath. *Phy Rev A*, 33(5):3628–3631, 1986.

[45] JP Ryckaert, G Ciccotti, and HJC Berendsen. Numerical-integration of the cartesian equations of motion of a system with constraints: Molecular dynamics of n-alkanes. *J. Comput. Phys.*, 23(3):327–341, 1977.

[46] David Dubbeldam, Gloria A. E. Oxford, Rajamani Krishna, Linda J. Broadbelt, and Randall Q. Snurr. Distance and angular holonomic constraints in molecular simulations. *J. Chem. Phys.*, 133(3), JUL 21 2010.

[47] Peter Eastman and Vijay S. Pande. Constant constraint matrix approximation: A robust, parallelizable constraint method for molecular simulations. *J. Chem. Theory Comput.*, 6(2):434–437, 2010.

[48] D.J. Evans and S Murad. Singularity free algorithm for molecular dynamics simulation of rigid polyatomics. *Molecular Physics*, 34(2):327–331, 1977.

[49] DLPOLY. http://www.ccp5.ac.uk/DL_POLY/, Nov 2010.

[50] S Plimpton. Fast parallel algorithms for short-range molecular-dynamics. *J. Comput. Phys.*, 117(1):1–19, March 1995.

[51] Trung Dac Nguyen and Sharon C. Glotzer. Reconfigurable assemblies of shape-changing nanorods. *ACS Nano*, 4(5):2585–2594, May 2010.

[52] David R. Heine, Matt K. Petersen, and Gary S. Grest. Effect of particle shape and charge on bulk rheology of nanoparticle suspensions. *J. Chem. Phys.*, 132 (18), May 2010.

[53] Mark A. Horsch, Zhenli Zhang, and Sharon C. Glotzer. Self-assembly of end-tethered nanorods in a neat system and role of block fractions and aspect ratio. *Soft Matter*, 6(5):945–954, 2010.

[54] Trung Dac Nguyen and Sharon C. Glotzer. Switchable helical structures formed by the hierarchical self-assembly of laterally tethered nanorods. *Small*, 5(18): 2092–2098, September 2009.

[55] Trung Dac Nguyen, Zhenli Zhang, and Sharon C. Glotzer. Molecular simulation study of self-assembly of tethered V–shaped nanoparticles. *J. Chem. Phys.*, 129 (24), December 2008.

[56] JA Elliott and AH Windle. A dissipative particle dynamics method for modeling the geometrical packing of filler particles in polymer composites. *J. Chem. Phys.*, 113(22):10367–10376, December 2000.

[57] Raymond Mountain. Solvation structure of ions in water. *Int. J. Thermophys.*, 28:536–543, 2007. ISSN 0195-928X.

[58] Scott D. Johnson, Raymond D. Mountain, and Paul H. E. Meijer. Simulation of C-60 through the plastic transition temperatures. *J. Chem. Phys.*, 103(3): 1106–1108, 1995.

[59] T. F. Miller, M. Eleftheriou, P. Pattnaik, A. Ndirango, D. Newns, and G. J. Martyna. Symplectic quaternion scheme for biophysical molecular dynamics. *J. Chem. Phys.*, 116(20):8649–8659, 2002.

[60] LAMMPS source code. http://lammps.sandia.gov/, Nov 2010.

[61] H. Kameraj, R. J. Low, and M. P. Neal. Time reversible and symplectic integrators for molecular dynamics simulations of rigid molecules. *J. Chem. Phys.*, 122:224114–30, 2003.

[62] A. Gaikwad and I. M. Toke. GPU based sparse grid technique for solving multidimensional options pricing pdes. In *Proceedings of the 2nd Workshop on High Performance Computational Finance*, WHPCF '09, pages 6:1–6:9, New York, NY, USA, 2009. ACM.

[63] J. C. Thibault and I. Senocak. Cuda implementation of a Navier-Stokes solver on multi-GPU desktop platforms for incompressible flows. *47th AIAA Aerospace Sciences Meeting. Orlando, FL*, January 2010.

[64] D. B. Kirk and W. W. Hwu. *Programming Massively Parallel Processors: A Hands-on Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2010. ISBN 0123814723, 9780123814722.

[65] http://www.nvidia.com/cudazone, http://developer.nvidia.com/getcuda (cuda toolkit).

[66] J. A. Anderson, R. Sknepnek, and A. Travesset. Design of polymer nanocomposites in solution by polymer functionalization. *Phys. Rev. E*, 82(2), August 2010.

[67] Yong Duan and Peter A. Kollman. Pathways to a Protein Folding Intermediate Observed in a 1-Microsecond Simulation in Aqueous Solution. *Science*, 282 (5389):740–744, 1998.

[68] Christopher D. Snow, Houbi Nguyen, Vijay S. Pande, and Martin Gruebele. Absolute comparison of simulated and experimental protein-folding dynamics. *Nature*, 420(6911):102–106, October 2002.

[69] Peter L. Freddolino, Christopher B. Harrison, Yanxin Liu, and Klaus Schulten. Challenges in protein-folding simulations. *Nat. Phys.*, 6, 2010.

[70] Joshua A. Anderson, Chris D. Lorenz, and A. Travesset. General purpose molecular dynamics simulations fully implemented on graphics processing units. *J. Comput. Phys.*, 227(10):5342–5359, May .

[71] J. A. van Meel, A. Arnold, D. Frenkel, S. F. Portegies Zwart, and R. G. Belleman. Harvesting graphics power for MD simulations. *Molecular Simulation*, 34 (3):259–266, 2008.

[72] John E. Stone, James C. Phillips, Peter L. Freddolino, David J. Hardy, Leonardo G. Trabuco, and Klaus Schulten. Accelerating molecular modeling applications with graphics processors. *J. Comput. Chem.*, 28(16):2618–2640, December 2007.

[73] HOOMD-blue. http://codeblue.umich.edu/hoomd-blue/, Nov 2010.

[74] L. Martínez, R. Andrade, E. G. Birgin, and J. M. Martínez. PACKMOL: a package for building initial configurations for molecular dynamics simulations. *Journal of computational chemistry*, 30(13):2157–2164, October 2009.

[75] VMD is developed with NIH support by the Theoretical and Computational Biophysics group at the Beckman Institute, University of Illinois at Urbana-Champaign.

[76] John Stone. An efficient library for parallel ray tracing and animation. Master's thesis, Computer Science Department, University of Missouri-Rolla, April 1998.

[77] Erik Bitzek, Pekka Koskinen, Franz Gaehler, Michael Moseler, and Peter Gumbsch. Structural relaxation made simple. *Phys. Rev. Lett.*, 97(17), October 2006.

[78] P. J. Hoogerbrugge and J. M. V. A. Koelman. Simulating microscopic hydrodynamic phenomena with dissipative particle dynamics. *Europhysics Letters*, 19:155, 1992.

[79] J. M. V. A. Koelman and P. J. Hoogerbrugge. Dynamic simulations of hard-sphere suspensions under steady shear. *Europhysics Letters*, 21:363, 1993.

[80] K. Huang, C. Lin, H. Tsao, and Y. Sheng. The interactions between surfactants and vesicles: Dissipative particle dynamics. *J. Chem. Phys.*, 130(24):245101, 2009.

[81] M. Kenward and K. D. Dorfman. Brownian dynamics simulations of single-stranded DNA hairpins. *J. Chem. Phys.*, 130(9):095101, 2009.

[82] C. Singh, A. M. Jackson, F. Stellacci, and S. C. Glotzer. Exploiting substrate stress to modify nanoscale sam patterns. *J. Am. Chem. Soc.*, 131(45):16377–16379, 2009.

[83] C. L. Phillips and S. C. Glotzer. Effect of nanoparticle polydispersity on the self assembly of polymer tethered nanospheres. *preprint*.

[84] L. Gu, S. Xu, Z. Sun, and J. T. Wang. Brownian dynamics simulation of the crystallization dynamics of charged colloidal particles. *J. Colloid Interface Sci.*, 350(2):409 – 416, 2010. ISSN 0021-9797.

[85] Steve Worley. private communication, and *Saru* code package.

[86] David Wheeler and Roger Needham. TEA, a tiny encryption algorithm. pages 97–110. Springer-Verlag, 1995.

[87] F. Zafar, A. Curtis, and M. Olano. Gpu random numbers via the tiny encryption algorithm. *HPG 2010: Proceedings of the ACM SIGGRAPH/Eurographics Symposium on High Performance Graphics*, June 2009.

[88] D. E. Knuth. *The art of computer programming, volume 2 (3rd ed.): seminumerical algorithms*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1997.

[89] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1992.

[90] P. L'Ecuyer and R. Simard. Testu01: A C library for empirical testing of random number generators. *ACM Trans. Math. Softw.*, 33(4):22, 2007. ISSN 0098-3500. doi: http://doi.acm.org/10.1145/1268776.1268777.

[91] G. Marsaglia, A. Zaman, and W. W. Tsang. Toward a universal random number generator. *Statistics & Probability Letters*, 9(1):35 – 39, 1990.

[92] M. Matsumoto and T. Nishimura. Dynamic creation of pseudorandom number generators. In *Monte Carlo and Quasi-Monte Carlo Methods*, pages 56–69. Springer, 1998.

[93] B. L. Holian, O. E. Percus, T. T. Warnock, and P. A. Whitlock. Pseudorandom number generator for massively parallel molecular-dynamics simulations. *Phys. Rev. E*, 50(2):1607–1615, Aug 1994.

[94] H. Nguyen. *GPU Gems 3*. Addison-Wesley Professional, 2007.

[95] W. B. Langdon. A fast high quality pseudo random number generator for graphics processing units. In *In IEEE Congress on Evolutionary Computation, 2008. CEC 2008*, pages 459–465, 2008.

[96] A. Zhmurov, K. Rybnikov, Y. Kholodov, and V. Barsegov. Generation of random numbers on graphics processors: Forced indentation in silico of the bacteriophage hk97. *The Journal of Physical Chemistry B*, 115(18):5278–5288, 2011.

[97] S. Tzeng and L. Wei. Parallel white noise generation on a gpu via cryptographic hash. *I3D 2008, Association for Computing Machinery, Inc.*, October 2007.

[98] W. Liu, B. Schmidt, G. Voss, and W. Müller-Wittig. Accelerating molecular dynamics simulations using graphics processing units with cuda. *Comput. Phys. Commun.*, 179(9):634 – 641, 2008.

[99] J. Davis, A. Ozsoy, S. Patel, and M. Taufer. Towards large-scale molecular dynamics simulations on graphics processors. In Sanguthevar Rajasekaran, editor, *Bioinformatics and Computational Biology*, volume 5462 of *Lecture Notes in Computer Science*, pages 176–186. Springer Berlin / Heidelberg, 2009.

[100] D. C. Rapaport. Role of reversibility in viral capsid growth: A paradigm for self-assembly. *Phys. Rev. Lett.*, 101(18):186101, Oct 2008.

[101] A. Sunarso, T. Tsuji, and S. Chono. GPU-accelerated molecular dynamics simulation for study of liquid crystalline flows. *J. Comput. Phys.*, 229(15):5486 – 5497, 2010.

[102] M. S. Friedrichs, P. Eastman, V. Vaidyanathan, M. Houston, S. Legrand, A. L. Beberg, D. L. Ensign, C. M. Bruns, and V. S. Pande. Accelerating molecular dynamic simulation on graphics processing units. *J. Comput. Chem.*, 30(6): 864–872, April 2009.

[103] P. Eastman and V. S. Pande. Efficient nonbonded interactions for molecular dynamics on a graphics processing unit. *J. Comput. Chem.*, 31(6):1268–1272, October 2010.

[104] J. E. Stone, D. J. Hardy, I. S. Ufimtsev, and K. Schulten. GPU-accelerated molecular modeling coming of age. *J. of Molecular Graphics and Modelling*, 29 (2):116 – 125, 2010.

[105] D. L. Ermak and J. A. McCammon. Brownian dynamics with hydrodynamic interactions. *J. Chem. Phys.*, 69(4):1352–1360, 1978.

[106] Rémi Lespiat, Sylvie Cohen-Addad, and Reinhard Höhler. Jamming and flow of random-close-packed spherical bubbles: An analogy with granular materials. *Phys. Rev. Lett.*, 106(14):148302, Apr 2011.

[107] Hugo Jacquin, Ludovic Berthier, and Francesco Zamponi. Microscopic mean-field theory of the jamming transition. *Phys. Rev. Lett.*, 106(13):135702, Mar 2011.

[108] Cang Zhao, Kaiwen Tian, and Ning Xu. New jamming scenario: From marginal jamming to deep jamming. *Phys. Rev. Lett.*, 106(12):125503, Mar 2011.

[109] A. Mughal, H. K. Chan, and D. Weaire. Phyllotactic description of hard sphere packing in cylindrical channels. *Phys. Rev. Lett.*, 106(11):115704, Mar 2011.

[110] Robert S. Hoy and Corey S. O'Hern. Minimal energy packings and collapse of sticky tangent hard-sphere polymers. *Phys. Rev. Lett.*, 105(6):068001, Aug 2010.

[111] Alexander Jaoshvili, Andria Esakia, Massimo Porrati, and Paul M. Chaikin. Experiments on the random packing of tetrahedral dice. *Phys. Rev. Lett.*, 104 (18):185501, May 2010.

[112] Randall D. Kamien and Andrea J. Liu. Why is random close packing reproducible? *Phys. Rev. Lett.*, 99(15):155501, Oct 2007.

[113] Aleksandar Donev, Frank H. Stillinger, and Salvatore Torquato. Do binary hard disks exhibit an ideal glass transition? *Phys. Rev. Lett.*, 96(22):225502, Jun 2006.

[114] Charles Radin and Lorenzo Sadun. Structure of the hard sphere solid. *Phys. Rev. Lett.*, 94(1):015502, Jan 2005.

[115] R. Mahmoodi Baram, H. J. Herrmann, and N. Rivier. Space-filling bearings in three dimensions. *Phys. Rev. Lett.*, 92(4):044301, Jan 2004.

[116] Tomaso Aste. Circle, sphere, and drop packings. *Phys. Rev. E*, 53(3):2571–2579, Mar 1996.

[117] S. Torquato. Reformulation of the covering and quantizer problems as ground states of interacting particles. *Phys. Rev. E*, 82(5):056109, Nov 2010.

[118] C. Messenger, R. Prix, and M. A. Papa. Random template banks and relaxed lattice coverings. *Phys. Rev. D*, 79(10):104017, May 2009.

[119] C. Anteneodo and W. A. M. Morgado. Critical scaling in standard biased random walks. *Phys. Rev. Lett.*, 99(18):180602, Nov 2007.

[120] K. R. Coutinho, M. D. Coutinho-Filho, M. A. F. Gomes, and A. M. Nemirovsky. Partial and random lattice covering times in two dimensions. *Phys. Rev. Lett.*, 72(24):3745–3749, Jun 1994.

[121] Alain Verberkmoes and Bernard Nienhuis. Triangular trimers on the triangular lattice: An exact solution. *Phys. Rev. Lett.*, 83(20):3986–3989, Nov 1999.

[122] ZL Zhang, MA Horsch, MH Lamm, and SC Glotzer. Tethered nano building blocks: Toward a conceptual framework for nanoparticle self-assembly. *Nano Letters*, 3(10):1341–1346, 2003.

[123] Ting Chen, Zhenli Zhang, and Sharon C. Glotzer. A precise packing sequence for self-assembled convex structures. *Proc Natl Acad Sci*, 104(3):717–722, January 2007.

[124] Tanja Schilling, Sander Pronk, Bela Mulder, and Daan Frenkel. Monte Carlo study of hard pentagons. *Phys. Rev. E*, 71:036138, Mar 2005.

[125] J. D. Bourland and Q. R. Wu. Use of shape for automated, optimized 3d radiosurgical treatment planning. In *Proceedings of the 4th International Conference on Visualization in Biomedical Computing*, pages 553–558, London, UK, 1996. Springer-Verlag.

[126] H. Blum and Roger N. Nagel. Shape description using weighted symmetric axis features. *Pattern Recognition*, 10(3):167–180, 1978.

[127] Harry Blum. A Transformation for Extracting New Descriptors of Shape. *Models for the Perception of Speech and Visual Form*, pages 362–380, 1967.

[128] D.J. Sheehy, C.G. Armstrong, and D.J. Robinson. Shape description by medial surface construction. *Visualization and Computer Graphics, IEEE Transactions on*, 2(1):62 –72, Mar 1996.

[129] Tim Culver, John Keyser, and Dinesh Manocha. Accurate computation of the medial axis of a polyhedron. In *Polyhedron, Fifth ACM Symposium on Solid Modeling*, pages 179–190, 1998.

[130] Tim Culver, John Keyser, and Dinesh Manocha. Exact computation of the medial axis of a polyhedron. *Comput. Aided Geom. Des.*, 21:65–98, January 2004.

[131] Josep Vilaplana. Computing the medial axis transform of polygonal objects by testing discs, 1996.

[132] F. P. Preparata. The medial axis of a simple polygon. *Proc. 6th Symp. Math. Foundations of Comput. Sci.*, pages 443–450, September 1977.

[133] H. Nebi Gürsoy and Nicholas M. Patrikalakis. An automatic coarse and fine surface mesh generation scheme based on medial axis transform: Part i algorithms *and* part ii implementatio. *Engineering with Computers*, 8:121–137, 1992.

[134] Suresh K. Generalization of the mid-element based dimensional reduction. *Journal of Computing and Information Science and Engineering*, 3, December 2003.

[135] Vakulenko A. Circles intersection, Technical report, 2004, from MATLAB Central File ExchangeFile id: 5313. http://www.mathworks.com/.

[136] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, 2006.

[137] Frederic Cazals, Harshad Kanhere, and Sebastien Loriot. Computing the volume of a union of balls: a certified algorithm. *ACM Transactions on Mathematical Software*, 38(1), 2011.

[138] private correspondence with Frederic Cazals.

[139] C Park, J Yoon, and EL Thomas. Enabling nanotechnology with self assembled block copolymer patterns. *Polymer*, 44(22):6725–6760, 2003.

[140] TP Lodge. Block copolymers: Past successes and future challenges. *Macromolecular Chemistry and Physics*, 204(2):265–273, 2003.

[141] MJ Fasolka and AM Mayes. Block copolymer thin films: Physics and applications. *Annual Review of Materials Research*, 31:323–355, 2001.

[142] CT Black, KW Guarini, R Ruiz, EM Sikorski, IV Babich, RL Sandstrom, and Y Zhang. Polymer self assembly in semiconductor microelectronics. *IBM Journal of Research and Development*, 51(5), 2007.

[143] MP Stoykovich and PF Nealey. Block copolymers and conventional lithography. *Materials Today*, 9(9):20–29, 2006.

[144] J Yoon, W Lee, and EL Thomas. Self-assembly of block copolymers for photonic-bandgap materials. *MRS Bulletin*, 30(10):721–726, 2005.

[145] Y Fink, AM Urbas, MG Bawendi, JD Joannopoulos, and EL Thomas. Block copolymers as photonic bandgap materials. *Journal of Lightwave Technology*, 17(11):1963–1969, 1999.

[146] B Jeong, YH Bae, DS Lee, and SW Kim. Biodegradable block copolymers as injectable drug-delivery systems. *Nature*, 388(6645):860–862, 1997.

[147] GS Kwon. Block copolymer micelles as drug delivery systems. *Advanced Drug Delivery Reviews*, 54(2):167–167, 2002.

[148] SC Glotzer and MJ Solomon. Anisotropy of building blocks and their assembly into complex structures. *Nature Materials*, 6:557–562, 2007.

[149] Sebastian Westenhoff and Nicholas A. Kotov. Quantum dot on a rope. *Journal of the American Chemical Society*, 124(11):2448–2449, 2002.

[150] Kie-Moon Sung, David W. Mosley, Beau R. Peelle, Shuguang Zhang, and Joseph M. Jacobson. Synthesis of monofunctionalized gold nanoparticles by fmoc solid-phase reactions. *Journal of the American Chemical Society*, 126(16): 5064–5065, 2004.

[151] F Huo, AKR Lytton-Jea, and CA Mirkin. Asymmetric functionalization of nanoparticles based on thermally addressable dna interconnects. *Advanced Materials*, 18(17):2304–2306, 2006.

[152] JG Worden, AW Shaffer, and Q Huo. Controlled functionalization of gold nanoparticles through a solid phase synthesis approach. *Chemical Communications*, (5):518519, 2004.

[153] GA DeVries, M Brunnbauer, Y Hu, AM Jackson, B. Long, B. T. Neltner, O. Uzun, B. H. Wunsch, and F. Stellacci. Divalent metal nanoparticles. *Science*, 315(5810):358–361, 2007.

[154] FA Aldaye and HF Sleiman. Dynamic dna templates for discrete gold nanoparticle assemblies: Control of geometry, modularity, write/erase and structural switching. *Journal of the American Chemical Society*, 129(14):4130–4131, 2007.

[155] CR Iacovella and SC Glotzer. Local ordering of polymer-tethered nanospheres and nanorods. *Journal of Chemical Physics*, 129, 2008.

[156] FJ Martinez-Veracoechea and FA Escobedo. Lattice monte carlo simulations of the gyroid phase in monodisperse and bidisperse block copolymer systems. *Macromolecules*, 38(20):8522–8531, 2005.

[157] DA Hajduk, PE Harper, SM Gruner, CC Honeker, G Kim, EL Thomas, and LJ Fetters. The gyroid - a new equilibrium morphology in a weakly segregated diblock copolymers. *Macromolecules*, 27(15):4063–4075, 1994.

[158] MA Horsch, ZL Zhang, CR Iacovella, and SC Glotzer. Hydrodynamics and microphase ordering in block copolymers: Are hydrodynamics required for ordered phases with periodicity in more than one dimension? *Journal of Chemical Physics*, 121(22):11455–11462, 2004.

[159] MM Maye, WX Zheng, FL Leibowitz, NK Ly, and CJ Zhong. Heating-induced evolution of thiolate-encapsulated gold nanoparticles: A strategy for size and shape manipulations. *Langmuir*, 16(2):490–497, 2000.

[160] MY Ge, HP Wu, L Niu, JF Liu, SY Chen, PY Shen, YW Zeng, YW Wang, GQ Zhang, and JZ Jiang. Nanostructured ZnO: from monodisperse nanoparticles to nanorods. *Journal of Crystal Growth*, 305(1):162–166, 2007.

[161] WL Pei, S Kakibe, I Ohta, and M Takahashi. Controlled monodisperse Fe nanoparticles synthesized by chemical method. *IEEE Transactions On Magnetics*, 41(10):3391–3393, 2005.

[162] P G Bolhuis and D A Kofke. Monte carlo study of freezing of polydisperse hard spheres. *Phys Rev E*, 54(1):634–643, 1996.

[163] PN Pusey. The effect of polydispersity of the crystalization of hard spherical colloids. *Journal de Physique*, 48:709–712, 1987.

[164] PN Pusey. Colloidal suspensions. In Hansen JP, Levesque D, and Zinn-Justin J, editors, *Liquids, Freezing, and Glass Transition*, pages 763–942.

[165] F M van der Koolj, K Kassapidou, and H N W Lekkerkerker. Liquid crystal phase transition in suspensions of polydisperse plate-like particles. *Nature*, 406: 868–871, 2000.

[166] RP Sear. Phase separation and crystallisation of polydisperse hard spheres. *Europhys. Lett.*, 44(4):531–535, 1998.

[167] M A Bates and D Frenkel. Influence of polydispersity on the phase behavior of colloidal liquid crystals: A monte carlo simulation study. *J. Chem. Phys*, 109, 1998.

[168] M Maldovan, AM Urbas, N Yulfa, WC Carter, and EL Thomas. Photonic properties of bicontinuous cubic microphases. *Phys Rev B*, 65(16), 2002.

[169] AJ Schultz. *Modeling and Computer Simulation of Block Copolymer/Nanoparticle Composites*. PhD thesis, North Carolina State University, 2003.

[170] M Nonomura, K Yamada, and T Ohta. Formation and stability of double gyroid in microphase-separated diblock copolymers. *Journal of Physics - Condensed Matter*, 15(26):L423–L430, 2003.

[171] PJ Steinhardt, DR Nelson, and M Ronchetti. Bond-orientational order in liquids and glasses. *Phys Rev B*, 28(2):784–805, 1983.

[172] DJ Wales and JPK Doye. Global optimization by basin-hopping and the lowest energy structures of lennard-jones clusters containing up to 110 atoms. *Journal of Physical Chemistry A*, 101(28):5111–5116, 1997.

[173] Ludger Santen and Werner Krauth. Liquid, glass and crystal in two-dimensional hard disks. 2001.

[174] David A. Kofke and Peter G. Bolhuis. Freezing of polydisperse hard spheres. *Phys. Rev. E*, 59(1):618–622, Jan 1999.

[175] Stefan Auer and Daan Frenkel. Suppression of crystal nucleation in polydisperse colloids due to increase of the surface free energy. *Nature*, 413(6857):711–713, October 2001.

[176] Paul Bartlett. Freezing in polydisperse colloidal suspensions. *Journal of Physics: Condensed Matter*, 12(8A):A275–A280, 2000.

[177] M. Reza Sadr-Lahijany, Purusattam Ray, and H. Eugene Stanley. Dispersity-driven melting transition in two-dimensional solids. *Phys. Rev. Lett.*, 79(17): 3206–3209, Oct 1997.

[178] Sander Pronk and Daan Frenkel. Melting of polydisperse hard disks. *Phys. Rev. E*, 69(6):066123, Jun 2004.

[179] Stephen Martin, Gary Bryant, and William van Megen. Observation of a smecticlike crystalline structure in polydisperse colloids. *Phys. Rev. Lett.*, 90(25): 255702, Jun 2003.

[180] S. Martin, G. Bryant, and W. van Megen. Crystallization kinetics of polydisperse colloidal hard spheres: Experimental evidence for local fractionation. *Phys. Rev. E*, 67(6):061405, Jun 2003.

[181] S Cozzini and M Ronchetti. Local icosahedral structures in binary-alloy clusters from molecular-dynamics simulation. *Phys Rev B*, 53(18):12040–12049, 1996.

[182] D He, NN Ekere, and L Cai. Computer simulation of random packing of unequal particles. *Phys Rev E*, 60(6), 1999.

[183] N N Medvedev, A Geiger, and W Brostow. Distinguishing liquids from amorphous solids: Percolation analysis on the voronoi network. *J. Chem. Phys.*, 93 (11):8337–8342, 1990.

[184] JC Gil Montoro and JLJ Abascal. The Voronoi polyhedra as tools for structure determination in simple disordered systems. *J. Phys. Chem.*, 97:4211–4215, 1993.

[185] VS Kumar and V Kumaran. Voronoi neighbor statistics of hard-disks and hard-spheres. *J. Phys. Chem.*, 123(7):074502, 2005.

[186] V S Kumar and V Kumaran. Voronoi cell volume distribution and configurational entropy of hard-spheres. *J. Chem. Phys.*, 123(11):114501, 2005.

[187] JT Fern, DJ Keffer, and WV Steele. Measuring coexisting densities from a two-phase molecular dynamics simulation by voronoi tessellations. *J. Phys. Chem. B*, 111(13):3469–3475, 2007.

[188] JT Fern, DJ Keffer, and WV Steele. Vapor-liquid equilibrium of ethanol by molecular dynamics simulation and voronoi tessellation. *J. Phys. Chem. B*, 111 (46):13278–13286, 2007.

[189] FB Usabiaga and D Duque. Applications of computational geometry to the molecular simulation of interfaces. *Physical Review E*, 79(4):046709, 2009.

[190] FW Starr, S Sastry, JF Douglas, and SC Glotzer. What do we learn from the local geometry of glass-forming liquids? *Phys. Rev. Lett.*, 89(12):125501, Aug 2002.

[191] L Leibler. Theory of microphase separation in block copolymers. *Macromolecules*, 13(6), 1980.

[192] IW Hamley, editor. *Developments in Block Copolymer Science and Technology*. Wiley, New York, 2004.

[193] IW Hamley. *The Physics of Block Copolymers*. Oxford University Press, Oxford, 1999.

[194] A Jayaraman and KS Schweizer. Effective interactions, structure, and phase behavior of lightly tethered nanoparticles in polymer melts. *Macromolecules*, 41(23):9430–9438, 2008.

[195] CH Rycroft. *Multiscale Modeling in Granular Flow*. PhD thesis, Massachusetts Institute of Technology, 2007.

[196] CA Lambert, LH Radzilowski, and EL Thomas. Triply periodic level surfaces as models for cubic tricontinuous block copolymer morphologies. *Philosophical Transactions of the Royal Society A*, 354(1715):2009–2023, 1996.

[197] T Aste and D Weaire. *The Pursuit of Perfect Packing*. Institute of Physics Publishing, 2000.

[198] A Gervois, P Richard, L Oger, and JP Troadec. A model of binary assemblies of spheres. *The European Physical Journal E - Soft Matter*, 6(4):295–303, 2001.

[199] NN Medvedev, VP Voloshin, VA Luchnikov, and ML Gavrilova. Algorithm for three-dimensional voronoi s-network. *Journal of Computational Chemistry*, 27 (14):1676–1692, 2006.

[200] EB Saff and ABJ Kuijlaars. Distributing many points on a sphere. *Mathematical Intelligencer*, 19(1):5–11, 1997.

[201] Stefano Sacanna and David J. Pine. Shape-anisotropic colloids: Building blocks for complex assemblies. *Current Opinion in Colloid & Interface Science*, January 2011.

[202] Matthew R. Jones, Robert J. Macfarlane, Byeongdu Lee, Jian Zhang, Kaylie L. Young, Andrew J. Senesi, and Chad A. Mirkin. Dna-nanoparticle superlattices formed from anisotropic building blocks. *Nature Materials*, 9:913–917, 2010.

[203] Szilard N. Fejer, Dwaipayan Chakrabarti, and David J. Wales. Emergent complexity from simple anisotropic building blocks: Shells, tubes, and spirals. *ACS Nano*, 4(1):219–228, 2010.

[204] Szilard N. Fejer, Dwaipayan Chakrabarti, and David J. Wales. Self-assembly of anisotropic particles. *Soft Matter*, 7:3553–3564, 2011.

[205] Alexander J. Williamson, Alex W. Wilber, Jonathan P. K. Doye, and Ard A. Louis. Templated self-assembly of patchy particles. *Soft Matter*, 7:3423–3431, 2011.

[206] E Jankowski and SC Glotzer. Calculation of partition functions for the self-assembly of patchy particles. *Journal of Physical Chemistry B*, 2011.

[207] Francesco Sciortino and Emanuela Zaccarelli. Reversible gels of patchy particles. *Current Opinion in Solid State and Materials Science*, In Press, Uncorrected Proof:–, 2011.

[208] Emanuela Bianchi, Ronald Blaak, and Christos N. Likos. Patchy colloids: state of the art and perspectives. *Phys. Chem. Chem. Phys.*, 13:6397–6410, 2011.

[209] Young-Sang Cho, Gi-Ra Yi, Shin-Hyun Kim, David J. Pine, and Seung-Man Yang. Colloidal clusters of microspheres from water-in-oil emulsions. *Chemistry of Materials*, 17(20):5006–5013, 2005.

[210] Young-Sang Cho, Gi-Ra Yi, Jong-Min Lim, Shin-Hyun Kim, Vinothan N. Manoharan, David J. Pine, and Seung-Man Yang. Self-organization of bidisperse colloids in water droplets. *Journal of the American Chemical Society*, 127 (45):15968–15975, 2005.

[211] Young-Sang Cho, Gi-Ra Yi, Shin-Hyun Kim, Mark T. Elsesser, Dana R. Breed, and Seung-Man Yang. Homogeneous and heterogeneous binary colloidal clusters formed by evaporation-induced self-assembly inside droplets. *Journal of Colloid and Interface Science*, 318(1):124 – 133, 2008.

[212] Eva G. Noya, Carlos Vega, Jonathan P. K. Doye, and Ard A. Louis. The stability of a crystal with diamond structure for patchy particles with tetrahedral symmetry. *The Journal of Chemical Phyics*, 132(23):234511, 2010.

[213] Fejes Tóth. *Regular Figures*. The Macmillan Company, 1964.

[214] Theodor William Melnyk, Osvald Knop, and William Robert Smith. Extremal arrangements of points and unit charges on a sphere: equilibrium configurations revisited. *Canadian Journal of Chemistry*, 55(10):1745–1761, 1977.

[215] L. L. Whyte. Unique arrangements of points on a sphere. *The American Mathematical Monthly*, 59(9):pp. 606–611, 1952.

[216] J. R. Edmundson. The distribution of point charges on a unit sphere. *Acta Cryst.*, A48:60–69, 1992.

[217] H. Cohn, Y. Jiao, A. Kumar, and S. Torquato. Rigidity of spherical codes. *ArXiv e-prints*, February 2011.

[218] T. Tarnai and Gáspár Zs. Improved packing of equal circles on a sphere and rigidity of its graph. *Mathematical Proceedings of the Cambridge Philosphical Society*, 93:191–218, 1983.

[219] J. A. Barker and D. Henderson. Perturbation Theory and Equation of State for Fluids. II. A Successful Theory of Liquids. *Journal of Chemical Physics*, 47: 4714–4721, December 1967.

[220] David J. McGinty. Vapor phase homogeneous nucleation and the thermodynamic properties of small clusters of argon atoms. *J. Chem. Phys.*, 55(2): 580–588, 1971.

[221] Guangnan Meng, Natalie Arkus, Michael P. Brenner, and Vinothan N. Manoharan. The free-energy landscape of clusters of attractive hard spheres. *Science*, 327(5965):560–563, 2010.

[222] Amir Haji-Akbari, Michael Engel, and Sharon C. Glotzer. Phase diagram of hard tetrahedra. *The Journal of Chemical Physics*, 135(19):194101, 2011.

[223] Eric R. Weeks and D. A. Weitz. Subdiffusion and the cage effect studied near the colloidal glass transition. *Chemical Physics*, 284(1-2):361 – 367, 2002.

[224] Henry Cohn and Abhinav Kumar. Universally optimal distribution of points on spheres. *J.AMER.MATH.SOC.*, 20:99, 2007.

[225] Brandon Ballinger, Grigoriy Blekherman, Henry Cohn, Noah Giansiracusa, Elizabeth Kelly, and Achill Schürmann. Experimental study of energy-minimizing point configurations on spheres. *Experimental Mathematics*, 18(3): 257–283, 2009.

[226] H. Cohn and A. Kumar. Algorithmic design of self-assembling structures. *ArXiv e-prints*, June 2009.

[227] GAP. *GAP – Groups, Algorithms, and Programming, Version 4.4.12*. The GAP Group, 2008. URL `(http://www.gap-system.org)`.