EASAL: Efficient Atlasing, Analysis and Search of Molecular Assembly Landscapes

Meera Sitharam∗†, Aysegul Ozkan∗, James Pence, Jorg Peters∗

Abstract
To elucidate the structure of assembly configuration spaces, the EASAL software combines classical concepts, such as stratifications of semialgebraic sets, with recent algorithms for efficiently realizing geometric constraint systems, and theoretical advances concerning convex parametrization: in contrast to folding configuration spaces, most regions of assembly and packing configurations admit a convex parametrization. This allows for a novel, efficient and intuitive representation of configuration spaces so that the corresponding atlas can be efficiently generated and sampled. This paper describes the approach, theory, structure and algorithms underlying EASAL and outlines its use for generating atlases of dimeric assemblies of the AAV2 coat protein, and alpha helix packing in transmembrane proteins.

1 Introduction
Efficient and intuitive description and prediction of the geometric structure and properties of high-dimensional molecular assembly (packing or docking) configuration spaces is a longstanding challenge. Since assembly is entropy-driven, analysis, search, sampling and visualization of spaces is geared towards determining the configurational entropy and towards isolating the intermolecular interactions that are crucial for successful assembly pathways. These, in turn, shed light on robust and spontaneous - but poorly understood - supramolecular and macromolecular self-assembly processes such as helix packing, viral self-assembly, protein crystallization, prion aggregation, ligand and drug docking etc.. This paper develops the theory, algorithms and prototype software implementation of EASAL, a framework for efficient atlas generation, sampling, searching, analysis and visualization for molecular assembly configuration spaces.

Contributions and Novelty. EASAL’s algorithms and software have been designed to provide the following features.
• Intuitive and explicit relationships between the input molecular data and the geometric properties of the configuration space (note the distinction between configuration space and Cartesian realizations);
• Quantitative accuracy guarantees derivable from the input data, including running time estimates;
• Flexible down-scaling to lower refinement, decreasing effort while preserving key features of the configuration space structure (such as lower dimensional boundaries that often include key regions of the configuration space);

∗CISE department, University of Florida, CSE Bldg, Gainesville, FL 32611-6120; phone - 3523921200; fax - 3523921220. This research was supported in part by NSF Mathematical Biology Grant DMS0714912, a University of Florida computational biology seed grant and NSF CCF-1117695.
†Corresponding author email: sitharam@cise.ufl.edu
• Intuitive visualization, GUI and other functionalities for the biophysics, biochemistry or structural biology users;
• Computational efficiency. Efficiency here means: repeated sampling is avoided and stays within or optimally close to the (feasible) configuration space. This makes EASAL effective on terabyte laptops, without data compression.

Theoretic and Algorithmic Contributions The new contributions are built on a novel combination of classical concepts such as strafication of semialgebraic sets and recent theoretical and algorithmic results on: (1) So called convex Cayley configuration spaces with efficient computable bounds and (2) decomposition of geometric constraint systems and optimizing the algebraic complexity of solving or realizing them.

Specifically, EASAL takes advantage of a key, new observation that assembly (but not structure determination or folding) admits configuration spaces whose regions can be parameterized over a convex domain, for example in terms of intermolecular distances. These are called convex Cayley parametrizations. In addition EASAL builds on to optimize the algebraic complexity of the polynomial systems that need to be solved for realization, i.e., to convert a parameterized Cayley configuration into its Cartesian realizations. Finally, EASAL leverages algebraic-numeric methods for capturing all real solutions of multivariate polynomial systems.

EASAL organizes the configuration space into regions of various dimensions, called active constraint regions, where sets of constraints are exactly met. This stratification by containment is akin to the Thom-Whitney stratification or road map of a semi-algebraic set and is an important ingredient for computing configurational entropy. See Fig. 2(a) For assembly configuration spaces, we observe and exploit the fact that each active constraint region has an inherent combinatorial structure, the active constraint graph, by which they can be uniquely labeled. The active constraint graph allows defining, for all convex configuration regions, efficiently computable charts that exactly parameterize the region, including boundaries. Stratification into active constraint region, together with chart of each region yields an atlas of the configuration space. Together these theoretical and algorithmic techniques make the assembly problem more tractable than typical conformational structure determination or folding problems. Specifically the atlas allows scalable and efficient sampling, search that enable entropy analysis and visualization of the configuration space.

Organization In Section 2 we contrast assembly from folding, discuss configurational entropy and the role of dynamics. Section 3 provides definitions and theory including the new observation that most regions of assembly configuration spaces have convex Cayley parametrizations specified by active constraint graphs. These graphs include the well-known class of partial 3-trees, or graphs with tree-width 3. The section also formally defines charts and atlas of configuration spaces. Section 4 presents the sampling the assembly configuration space of molecule parts and, for the case of two identical molecule parts with any number of atom markers, establishes correctness, and complexity. The Appendix lists the pseudocode. Section 5 summarizes how to realize configurations in Cartesian Space according to. Section 6 describes the architecture of the EASAL software and its visual user interface. Section 7 reports on experiments with assembly problems containing about 20 atom markers: two molecule parts with 10 atom markers each and three molecule parts with 6 atom markers each; and 6 molecule parts with 3 atom markers each. Section 8 discusses the choices made in EASAL and a number of recent or planned extensions.

Scope. We omit a discussion of how the configuration space atlas addresses configurational entropy and assembly pathways. Also a full account of the validation of EASAL, using experimental data from
virus assembly [17], is beyond the scope of the paper.

2 Background: assembly (not molecular conformation or folding), entropy and dynamics

Assembly. Molecular assembly and packing configuration spaces are defined by known intermolecular interactions between the constituent molecular units. Interactions include weak forces, hydrogen bonds, steric constraints, tethering constraints as well as global energy and symmetry constraints. The interactions, can, with some work, be represented as static, geometric constraints such as distance and angle intervals between geometric primitives that are used to represent molecular units. The composite of molecule parts and constraints is together called the assembly system (or packing system). Each point or configuration in the configuration space represents one Cartesian realizations of the entire molecular composite that is feasible, i.e., satisfies the given constraints and is therefore a solution to the assembly constraint system. The realization is specified by \(xyz\)-coordinates or by rotations and translations of the rigid molecule parts relative to each other, (modulo rotations and translations of the entire composite).

Assembly constraint systems in this paper are represented by static geometric constraints such as distance and angle intervals between atom markers (typically spheres or cylinders representing a geometrically significant collection of residues) in a molecule part. The behavior of large assemblies is typically analyzed by recursively decomposing and analyzing the configuration spaces of the smaller intermediate subassemblies and nucleations [31 [25, 2]. The largest of these decomposed intermediate systems have typically at most 10 molecular units, each with at most 50 atomic units. The total number of atom markers in these intermediate systems typically does not exceed 100. Since larger intermediate systems typically have internal symmetry and other constraints to satisfy, the dimension of an assembly configuration space obeying the constraints is expected to be at most 25, typically under 15.

Assembly versus molecular conformation and folding. In contrast to assembly, the configuration space for, say, a protein’s conformations or folding usually involves much smaller, but many more rigid units each representing essentially one residue or atomic unit. This makes the dimension of the configuration space significantly larger. Furthermore, instead of the sparse tree of tether constraints that holds together the rigid molecular units in an assembly, in a folding they are held together by “body-hinge” or exact distance constraints or relatively tight distance/angle intervals that model a protein backbone. These constraints form a relatively denser graph than a tree, often including cycles of distance constraints. However, even in the folding configuration space problem, weak forces are typically represented in a manner similar to the local atomic assembly constraints. Despite these differences, the most common methods so far for sampling or exploring configuration spaces do not exploit the special properties of assembly or packing configuration spaces: In Monte Carlo methods mixed with constraint resolution and/or energy minimization by gradient descent [15], or molecular dynamics with specially designed energy functions [4], the same methods are used for sampling and exploring molecular conformation or folding.

Configurational entropy and configuration space sampling. The configurational entropy of a collection of molecular units can be viewed as the volume of the configuration space weighted by the likelihood of occurrence during the relevant process. Together with energy values, this determines the free energy [16].
Molecular dynamics methods mixed with specially designed energy functions are commonly used for problems (i) and (ii). They are computationally very intensive and do not exploit the special properties of assembly or packing configuration spaces, as opposed to folding configuration spaces. If started from sufficiently many initial configurations and run long enough, they can sample and explore all likely regions of the configuration space and give a reasonable estimate of configurational entropy. However, the requirements (a), (b) and (d) are not met by such algorithms.

Other common methods for sampling or exploring configuration spaces, such as Monte Carlo methods mixed with constraint resolution and/or energy minimization by gradient descent are more efficient than molecular dynamics, but often go outside the feasible region and discard many samples, which hurts their efficiency. Furthermore, by their nature, these methods cannot guarantee uniform sampling of the configuration space and since they are not informed by true dynamics, repeat sampling is not consistent with more probable configurations. Overall, the requirements (a), (b), (c) are not met by such algorithms.

Neither of the above two methods differentiates between assembly/packing and conformation/folding spaces and hence neither leverages the special properties of the former.

**Dynamics.** A crucial issue to be addressed both for methods like [15] and ours presented here is that interactions, such as weak forces using static geometric constraints (distance or angle intervals) or spring-based energy constraints, are not directly informed by dynamics. Hence estimating this probability distribution of configurations requires particular care.

A simple, concrete example of this occurs in the representation of a weak attractive force between a pair of atomic units. The force is active only in configurations where the pair of atomic units are in close proximity. However, configurations that are not *proximal* for this pair are still feasible, hence only the steric or distance lower bound constraint is used to specify the overall configuration space. But, once a configuration is reached where this pair of atomic units is proximal, then the attractive force becomes active and the subsequent configurations will, with high probability, continue to be proximal for that pair, i.e., that pair of atomic units would satisfy a distance upper bound or a spring-type energy upper bound.

Another example is when two opposing repelling forces become active on the same atomic unit. Then their corresponding distance lower bound constraints are active (i.e., the bounds are met), and again, with high probability, those constraints will remain active. As another example, if there is a global constraint requiring the minimization of an energy function, the feasible configuration space can be modeled using a threshold on the energy function, but the probability distribution on the configuration space will depend on the energy function.

In all these and other such scenarios, in order to effectively estimate configurational entropy, specifically the probability distribution over the configuration space, it is necessary to form a stratification of the structure of the configuration space, including lower dimensional strata and regions where various specific subsets of constraints become active and where other precise conditions are met, such as proximality, symmetry or lower energy. Stratification of the configuration space also makes search and other analysis, sampling and visualization significantly more scalable and efficient.

### 3 Theory: Stratifications and Atlases of Assembly Configuration Spaces

Distinct from folding (or other systems that also lead to molecular configuration spaces) an *assembly constraint system* (or packing constraint system) consists of the following.
A collection of globally rigid molecule parts, each represented as the internal Cartesian coordinates of a collection of atom markers. An atom marker, in turn, is represented as a sphere (point, radius) or possibly cylinder (line segment, radius). Molecule parts usually approximate geometrically significant collections of residues (Figure 1).

We will assume identical molecule parts for ease of exposition since non-identical ones are not conceptually more difficult but increase nomenclature.

A set of intermolecular assembly constraints (or packing constraints), of three general types.

1. A local atomic assembly constraint is specified as a distance and/or angle bound or interval between a pair of atom markers in different molecule parts. Many of these assembly constraints are distance lower bounds representing steric constraints or repelling interactions. Sometimes, the configuration space is restricted to only those configurations where specific attractive interactions are known to be active, including hydrogen bonds or electrostatic or other weak forces. In this case, the corresponding constraints are specified as distance intervals.

2. A pairwise molecular tether constraint is specified between a pair of molecules by giving a set of pairwise distance upper bounds between pairs of atom markers, one in each of the molecule parts and stipulating that at least one of these distance upper bounds is met and a composite molecular tether constraint is specified between a composite of several molecules by stipulating that a tree of pairwise molecular tether constraints must be satisfied. See Fig. 1. Often the molecular tethers are specified as bi-tethers, i.e., at least 2 pairwise distance upper bounds are met between 4 participating atom markers in a pair of molecules. In this case, it is generally assumed that the two participating atom markers within each molecule part are usually in close proximity. See Fig. 1.

3. A global assembly constraint is specified as a bound on some (e.g. energy) function of (the
have the prescribed lengths. A graph is d-realizable if there exists a Euclidean space of some dimension and its edges represent equality or inequality distance constraints. To be precise, we need to define a distance constraint graph.

Next we characterize all convex Cayley Configuration Spaces. Next we introduce the stratification of an assembly configuration space. Stratified spaces have been studied extensively. Semi-algebraic sets, i.e. sets defined multivariate polynomial inequalities, admit a Thom-Whitney stratification with various useful properties. A version of stratified semi-algebraic sets, so-called roadmaps, have been studied since in the context of configuration spaces for motion-planning in robotics and other geometric algorithms. (Their more recent cousin, probabilistic roadmaps, are not relevant to this paper).

Our assembly configuration spaces are semi-algebraic sets whose variables are the coordinates of the atom markers internal to a molecule part. Since each local assembly constraint asserts a distance/angle value (equality) or a distance/angle interval (two inequalities) between the positions of the participating two atom markers, a configuration is a solution to a system of quadratic polynomial inequalities. Consider an assembly configuration space of an assembly of rigid molecule parts, defined by a system \( A \) of assembly constraints. The configuration space of the composite has dimension \( m \leq 6(k - 1) \), the number of internal degrees of freedom of the composite since a rigid object in Euclidean 3-space has 6 rotational and translational degrees of freedom. For \( k = 2 \), \( m \) is at most 6 and in the presence of a composite bi-tether constraint, it is at most 4.

A stratification of the configuration space \( A \) is a partition of the space into regions grouped into strata \( X_i \) of \( A \) that form a filtration \( \emptyset \subset X_0 \subset X_1 \subset \ldots \subset X_m = A \), \( m = 6(k - 1) \). Each \( X_i \) is a union of nonempty closed active constraint regions \( R_Q \) where \( m - i \) inequality constraints \( Q \subseteq A \) are active, meaning equality is attained and they are independent (cf. Fig. 2(a)). Each active constraint set \( Q \) is itself part of at least one, and possibly many, hence \( l \)-indexed, nested chains of the form \( \emptyset \subset Q_0 \subset Q_1 \subset \ldots \subset Q_{m-i} = Q \subset \ldots \subset Q_m \) (cf. Fig. 3(b)). These induce corresponding reverse nested chains of active constraint regions \( R_{Q_j} : \emptyset \subset R_{Q_{l_0}} \subset R_{Q_{l_{m-1}}} \subset \ldots \subset R_{Q_{l_{m-i}}} = R_Q \subset \ldots \subset R_{Q_l} \). Note that here for all \( l, j, R_{Q_{l_{m-j}}} \subset X_j \) is closed and \( j \) dimensional.

As shown in Figure 2(b), we represent the active constraint system by a graph with vertices representing the participating atom markers (at least 3 in each atom marker) and edges representing the active constraints between them. Between a pair of molecule parts, there are only a small number of possible active constraint graph isomorphism types (all have at most 12 vertices).

There could be regions of the stratification of dimension \( j \) whose number of active constraints exceeds \( 6(k - 1) - j \), i.e. the active constraint system is overconstrained, or whose active constraints are not all independent. Dependent constraints diminish the set of realizations. For entropy calculations, these regions should be tracked explicitly, but in the present paper, we do not consider these special regions in the stratification. Our regions are obtained by choosing any \( 6(k - 1) - j \) independent active constraints.

Convex Cayley Configuration Spaces. Next we characterize all convex parametrizable 3-d distance constraint graphs as 3-realizable graphs. The vertices of a 3-d distance constraint graph are points in 3 dimensional Euclidean space and its edges represent equality or inequality distance constraints. To be precise, we need to define \( d \)-realizable graphs. and the inherently convex Cayley configuration space for a distance constraint graph.

Definition 3.1 ([34]). A graph with an assignment of distance values to its edges is Euclidean realizable if there exists a Euclidean space of some dimension where the vertices can be positioned so that the edges have the prescribed lengths. A graph is \( d \)-realizable if, for every distance assignment that is Euclidean
Figure 2: (a) Strata of different dimensions, consisting of active constraint regions of the assembly constraint system shown in the inset. The nodes of the directed acyclic graph are the active constraint regions and the edges indicate containment in a parent region one dimension higher. (b) All possible well-constrained active constraint graphs for assembly of two molecular units. The active constraint graphs are all possible subgraphs.

realizable, the vertices can be positioned so that the edges have the prescribed lengths can be satisfied in $d$ dimensional Euclidean space.

**Definition 3.2** (inherently convex Cayley configuration space). Let $G = (V, E)$ be a distance constraint graph, $E = H \cup F$ any partition of its edges, $d_F$ any fixed value of distances associated with $F$, and $d_H$ any distance inequalities associated with $H$. Denote by $\Phi_H(G, F, d_F, d_H)$ the set of all possible values of squared-distances for $H$, attained by Cartesian realizations of the vertices of $G$ and satisfying the constraints $d_F$ and $d_H$. If $\Phi_H(G, F, d_F, d_H)$ is convex then $G$ has an inherently convex Cayley configuration space.

Each Cayley point (set of distances) in $\Phi_H(G, F, d_F, d_H)$ corresponds to at least one, but potentially many Cartesian realizations of the distance constraint system given by $G$ and $d_F, d_H$. If $G$ is generically well-constrained (sometimes called minimally rigid [14]) then, for every point in $\Phi_H(G, F, d_F, d_H)$, the corresponding set of Cartesian realizations is generically finite (cf. Fig. 3(a)).

Until Definition 3.6 for ease of exposition, we focus on the active constraint graph $G$, assuming that the active constraint system is the entire assembly constraint system. That is, we ignore the remaining assembly constraint system, the atom markers that are not part of $G$ and the inactive constraints that involve them.

The next definition (cf. Fig. 3(a) and 3(b)) connects convex Cayley configuration spaces to (parametrizing) active constraint regions of assembly configuration spaces.

**Definition 3.3** (charts of distance constraint systems). Assume a distance constraint graph $G_F = (V, F)$ can be extended by an edge-set $H$ to $G = (V, E = H \cup F)$ so that it has an inherently convex 3-d Cayley configuration space. Then the active constraint region $R_{G_F}$, parametrized by the squared-distance
Figure 3: (a) **Nested regions in the stratification** containing the 2-dim active constraint region shown at the center. To the left are parent regions of higher dimension containing it and to the right child regions of lower dimension contained in it. Each region is shown as a sweep of all the Cartesian realizations in it (the blue molecule part is fixed without loss of generality). Note that each region is itself decomposed into Cartesian realizations of different chirality, each shown as a separate sweep. (b) **Cartesian realizations** of the same Cayley point. That is all shown Cartesian realizations have identical Cayley parameter values. The active constraint graph as well as chosen parameters of the region are displayed directly on the realizations.

(Cayley) parameters associated with the edges $H$, is $\Phi_H(G, F, d_F, d_H)$ (Definition 3.2) and is guaranteed to be convex. If $G$ is additionally well-constrained, the parameterization is called an **exact convex chart** of the active constraint region $R_{G,F}$ in the parameters $H$.

**Observation 1.** Every Cayley point in the exact convex chart $\Phi_H(G, F, d_F, d_H)$ has at least 1 and generically at most finitely many Cartesian realizations in the region $R_{G,F}$. See Figure 3(a).

The next theorem characterizes active constraint graphs that admit an exact convex chart parametrization.

**Theorem 3.4.** [32] A 3-d distance constraint graph $G = (V, E)$ has an inherently convex Cayley configuration space if and only if it is 3-realizable.

The 3-realizable graphs have a forbidden-minor characterization [34]: they do neither contain subgraphs homeomorphic to the complete graph on 5 vertices nor to the complete tripartite graph on 2 vertices. A natural class of 3-realizable graphs (cf. Figure 2(b)), the partial 3-trees, occur often as active constraint graphs. Partial 3-trees avoid 2 additional minors and can be defined in a recursive way using so-called 3-sums. A **partial 3-tree** is obtained from a complete 3-tree by removing edges from a complete 3-tree. A complete 3-tree consists either of two complete 3-trees sharing a triangle or, alternatively, is built up from a triangle by adding, at each step, a new vertex edge-connected to the edges of a triangle. Fig. 4 shows a partial 3-tree.

The next theorem indicates how to choose the parameters to obtain an exact convex chart for an active constraint region corresponding to a partial 3-tree; and how to compute its description and bounds. The choice of parameters is crucial: the paper [32] gives elementary examples that illustrate how one
Figure 4: Partial 3-tree construction or recognition: Determine base tetrahedra. Connect vertices to the faces of base tetrahedra.

choice of parameters yields a convex chart while others give nonconvex or disconnected ones (cf. Fig. 5). The theorem below also states the descriptive complexity of the convex chart, the number of boundaries and how efficient it is to compute them. This permits efficient exploration of the corresponding active constraint region.

Theorem 3.5 ([32] partial 3-tree yields exact convex chart). If an active constraint graph $G_F = (V, F)$ is a partial 3-tree then, by adding edge set $H$ to give a complete 3-tree $G = (V, E = H \cup F)$, we obtain an exact convex chart $\Phi_H(G, F, d_F, d_H)$ in the parameters $H$ of the active constraint region $R_{G_F}$. The exact convex chart $\Phi_H(G, F, d_F, d_H)$ has a linear number of boundaries in $|G|$ that can be output as implicit quadratic polynomial equalities in linear time. If we fix the parameters in $H$ in sequence, their explicit bounds can be computed in quadratic time in $|G|$.

For two molecule parts, $G$ has most 12 vertices. Tighter bounds are given in [5].

Figure 5: Parameterizations in configuration space. Note that the non-convex shape displayed in (b) and (c) expresses that no choice of Cayley parameters can make the region convex.

Parameterization of the Cayley space. In assembly systems, as opposed to general molecular conformational folding systems, most $G$ are partial 3-trees and hence a with an efficient parameterization in the form of an exact convex chart (Fig. 3a). Moreover, we can obtain exact convex charts for all 3-realizable graphs $G$ that can be completed to be well-constrained, by the method of [21] for recognizing and extending $G$. For the remaining 3-realizable graphs, and the few non-3-realizable ones of Fig. 2(b), there are techniques to provide tight convex charts that are close approximations to exact charts (Fig. 3b). However, given the rare occurrence of these cases, the current EASAL-implementation uses the routine non3Explore to step uniformly through the ambient lower-dimensional space defined by each choice of the parent’s parameter values, rather than restricting itself to just the feasible parameter set (Fig. 3c). By refining the parent chart’s parameters we can still guarantee a complete sampling of the region by witness points. Section 4

Note 1. In the discussion of Cayley spaces above, we defined regions $R'_G$ for an active constraint graph $G$, intentionally ignoring the remainder of the assembly constraint system, namely atom markers not in $G$ and their constraints. Fig. 6 The true active constraint region $R_G$ is an arbitrary subset of $R'_G$. When
Figure 6: Each color represents a boundary for the child region. Region (b) is created by accumulating points from child regions after reparametrizing in the chart of (a). The sampling is more dense in the child’s own chart, hence the region in (b) is denser than the region in (a) and there are extra points in (b) which are not captured in (a). On the other hand, there is missing regions in (b), because (b) includes only boundary regions not interior regions.

Figure 7: Sampling in own chart vs parent chart. (a) The parent’s uniform sampling (in the parametrization of its chart) yields only the red points in the child’s chart. (b) The child’s configuration space view reveals three additional realizations to the parent’s red realization.
a constraint (edge $e$) not in $G$ becomes active (at a configuration $c$ in $R'_G$), $G \cup \{e\}$ defines a child active constraint region $R_{Gue}$ containing $c$. This new region belongs to the stratum of the assembly configuration space that is of one lower dimension (Definition 3) and defines within $R'_G$ a boundary of the smaller, true active constraint region $R_G$. see Algorithm 3 We can still choose the chart of $R'_G$ as tight convex chart for $R_G$, but now region $R_{Gue}$ has an exact or tight convex chart of its own. This is useful, for example, if $e$ represents an attractive force constraint. Then the configurations in the region $R_{Gue}$ can be weighted heavier in entropy computations. Also Fig. 7 cautions that providing a separate chart for each active constraint region can reveal additional realizations at the same level of sampling.

We now define the atlas (cf. Fig. 2(a) and 8).

**Figure 8:** Information in the atlas on nested chains involving one region, i.e. on those paths in the directed acyclic graph of the stratification containing the node at the center. Each region has its active constraints graph and chart shown next to it. All the 3-dimensional parent charts have the 2-dimensional child region highlighted. The 2-dimensional (exact, convex) chart has a hole of infeasible configurations (cut out by a constraint outside the active constraint graph), but the same hole does not appear when parameterized in any of the parent charts.

**Definition 3.6 (Atlas).** An atlas of an assembly configuration space is a representation of its stratification into active constraint regions. Each active constraint region is represented by its active constraint graph, its exact convex or its tight chart and the parameters used for obtaining the chart or by non3Explore.

**Cartesian realization.** Finally, the Cayley points of the atlas need to be converted to Cartesian realizations. If the active constraint graph is a partial 3-tree or a Henneberg 1 graph (a generalization of partial 3-trees), realization is straightforward. For others that are merely 3-realizable or the rare cases that are none of the above, EASAL uses a decomposition algorithm [9], an algorithm to optimize algebraic complexity of the recombination systems to be solved [33] and finally the subdivision-based algebraic system solver [13]. Theorem 3.7 asserts that, based on a combinatorial method for optimizing the algebraic complexity of constraint graph [33], the realizations of the configurations in the active constraint regions can be recovered efficiently from the charts in the atlas. Such Euclidean realizations of the subgraphs of a distance constraint graph of $G$ are a solution to a polynomial system $P_G$. For different parametrizations, the algebraic complexity, i.e., number of variables, equations and degree of $P_G$, can vary widely.
Theorem 3.7 (see [33]). Let $G$ be a well-constrained 3d distance constraint graph decomposed into well-constrained or minimally rigid subgraphs that are maximal in the sense that no well-constrained graph contains them except possibly $G$ itself. Let $P_G$ be the polynomial system for obtaining the Euclidean realizations of $G$ from the realizations of these subgraphs. There is an algorithm that runs in time linear in $|G|$ that optimizes the algebraic complexity of $P_G$ within a class of natural parametrizations.

4 The EASAL Algorithm and Formal Guarantees

4.1 The Algorithm

On input of an assembly (or packing) system, the deterministic EASAL algorithm outputs a visual, query-searchable stratification of the cartesian configuration space (Fig. 8); and, if required, it samples the space at a desired level of refinement. The algorithm captures, stores and labels the regions of the stratification of the configuration space by efficiently computing an atlas of charts that parameterize the regions. The regions of the atlas are stored as nodes of a directed acyclic graph, Fig. 2(b) whose edges represent immediate containment or reachability. Each region of the atlas is identified by a (small) activeConstraintGraph $G$. Newly computed regions are tested by AlreadySeen to ensure that they are not already present in the current stratification. Only if $G$ is new, is the region further explored, by default, depth first. New active constraints and regions are added one by one, checking the non-active constraints between the atom markers of the remaining molecule parts. For simplicity of exposition, we explain the algorithm for a molecular assembly with two molecule parts, traversing depth first. Extended features are listed in Section 8. Pseudocode of all routines is listed in the Appendix.

Algorithm Synopsis. StartAtlas generates the stratification and the atlas by calling, for each set of active constraints given by a biTether, the routine ExploreSubAtlas. Whenever its subroutines SearchExplore or SampleExplore discover a new active constraint (cf. Fig. 10), ExploreSubAtlas is called on the corresponding lower dimensional region. non3Explore is called for constraints that do not have a well-constrained 3-realizable completion. Otherwise SampleExplore finds new active constraints by uniform sampling while SearchExplore applies binary search to the parameters selected by GenerateConvexParameters. GenerateConvexParameters applies the convex parametrization theory from Section 3 to choose the parameters for an active constraint system. Then the resulting Cayley configuration space is convex, before collisions or other (e.g. angle) constraints are introduced. Both search and stepping are restricted
to the exact convex cover of the parametrized parent region of the current active constraint region. *ConvexChartBoundaries* returns the bounds of the region corresponding to active constraint graphs of partial 3-trees and a tight convex cover for other 3-trees. The *Realize* method used by both exploration modes optimizes algebraic complexity and captures all solutions as explained in Section 5.

### 4.2 Correctness and Complexity

The complexity of the algorithm depends both on the input (number of molecular units, their size, number of constraints, required level of refinement) and the output (number of atlas nodes).

**Proposition 4.1.** *EASAL* (Algorithm 1)

a. creates and explores only regions that contain at least one Cartesian realization.

b. For the given tolerance $\tau$, if the configuration space is connected, it creates and explores all regions that contain at least one Cartesian realization.

c. If a region and its child regions have exact convex charts, it discards a minimum number of configurations.

**Proof.**

a. The algorithm starts with feasible active constraint sets i.e. there exists a Cartesian realization, a witness. *SampleExplore* or *SearchExplore* or *non3Explore* create a new active constraint graph $G$ only if the new region is proven to be feasible within tolerance $\tau$ and $\text{AlreadySeen}(G)$ is false; otherwise the routine continues to sample, respectively search.

b. No regions are missed due to pruning: if $G$ is rejected as infeasible, none of its child regions could have been feasible since they correspond to supersets of constraints. Since the space is connected, *SampleExplore* or *SearchExplore* will explore the region using efficient parameterizations or *non3Explore* will sample the current (ambient) space sufficiently densely.

c. An exact convex chart yields feasible Cayley points for the current active constraint set. Exploration (binary search) is required to find any additional constraints that restrict the region, e.g. steric constraints that render a configuration infeasible. Configurations therefore only need to be discarded during the binary search in *SampleExplore* or *SearchExplore* which finds the (lower-dimensional) boundary of the restriction (to be recursively explored and returning the next feasible point in the current chart.)

**Theorem 4.2** (Complexity). Let $k$ denote the total number of molecular units, $N$ the number of atomic units per molecule. and $\rho$ the fixed ratio of feasible sample points to all points sampled.

1. The maximum number of nonempty initial active constraint regions in the atlas is $O(N^{k-1})$.  

---

Figure 10: Euclidean realizations (*top*) and activeConstraintGraph $G$ (*bottom*) as constraints are added one by one. White lines represent active constraints, colored lines parameters.
2. Each new node of the atlas requires \( O(kN^2) \) time.

3. The average time complexity of computing a feasible sample point is constant for fixed \( k \) and \( N \).

**Proof.** 1. StartAtlas generates only regions defined by bi-tethers. Let \( t(k) \) be the number of possible non-isomorphic trees of size \( k \). For \( k \) molecules with \( N \) atoms each, the number of possible bi-tethers is \( t(k)N^{k-1} \).

2. Only SampleExplore, SearchExplore and ExploreSubAtlas can establish a new node. The routine non3Explore only triggers addition of witness configurations, i.e. Cartesian realizations, to the atlas. The subroutine Realize has constant time complexity: if the active constraint graph \( G \) forms a 3-tree, it can be realized in time exponential in the size \( G \) which has a constant upper bound in terms of \( k \); if \( G \) is not a 3-tree, Proposition 3.7 applies. The routine NonactiveConstraintCheck compares, in the worst case, all atom marker pairs, i.e. has time complexity \( O(N^2) \). Therefore, given the stepSize \( h \) and tolerance \( \tau \), the binary search in either SampleExplore or SearchExplore takes \( O(\log(h/\tau)N^2) \) time: in each binary refinement Realize and NonactiveConstraintCheck take at most \( O(N^2) \) time, \( kN^2 \) with \( h \) being halved, the loop ends when \( h < \tau \).

3. AlreadySeen prevents re-exploring \( G \). The work for each sample point is therefore the amortized cost of ConvexChartBoundaries in ExploreSubAtlas and of Realize and NonactiveConstraintCheck which were already characterized in 2. By Theorem 3.5 the bounds of convex charts of active constraint regions are computed in time quadratic in the size of the active constraint graph \( G \). (If \( k = 2 \) this time is constant.) \qed

### 5 Realizations in Euclidean Space

Obtaining a *realization* of a collection \( C := \{c_1, \ldots, c_n\} \) of rigid bodies means fixing a coordinate system, say that of \( c_1 \), and repositioning \( c_2, \ldots, c_n \) in the coordinate system of \( c_1 \) in such a way that distances are satisfied. Concretely, let \( x_{i,c_j} \in \mathbb{R}^3 \) be the coordinates of a point \( v_i \) in \( c_j \)'s *local* coordinate system. Given these local coordinates, we want to determine a rigid motions, i.e. translations \( t_j \in \mathbb{R}^3 \) and the six free parameters of a real symmetric \( 3 \times 3 \) matrix \( M_j \), representing the composition of three rotations, so that for all shared points \( v_i \) and \( j, k \neq 1 \),

\[
\begin{align*}
x_{i,c_1} &= M_j x_{i,c_j} + t_j, \\
M_j x_{i,c_j} + t_j &= M_k x_{i,c_k} + t_k,
\end{align*}
\]

and the distance constraints are satisfied in the coordinate system of \( c_1 \). That is we need to solve a system of polynomial equations to switch from configurations parameterized in terms of distances, to geometric realizations of molecule parts positioned next to each other.

Since all known solvers slow drastically with the increase in the number of equations, it is imperative to apply a form of *non-linear partial elimination* to the initial formulation. Such an elimination, generalizing tree-based elimination of chains or cycles of molecular bonds or articulated robotic links [26, 36, 6, 7], was proposed in [33]: The *Optimal Incidence Tree Algorithm* determines a partial elimination ordering that minimizes first the number of variables and then the degree in the rationally parametrized recombination system. Identifying same vertices in different coordinate systems and reformulating the remaining degrees of freedom in terms of the parameters of a rational, namely the stereographic map, the algorithm reduces the system (1) for discovering all allowable positions of two molecule parts, to at most four (generically three) unknowns. But, in its general implementation [33], this elimination takes ca. two minutes per
system. Fortunately, in our particular application, all possible structure of the systems are known beforehand and can be treated specifically without recourse to general pre-processing. We therefore converted the symbolic computations to C++ code and specialized and optimized the algorithm to the cases listed in Fig. 2(b). This reduces the conversion to at most two seconds.

It remains to solve the resulting polynomial systems, here of size $4 \times 4$ or less. Since we want to be able to exhaustively enumerate all solutions, applying (damped) Newton’s method to find all roots is not a safe option. Also, since we have thousands of systems to deal with, Groebner base computations do not succeed and neither do, in our experience, homotopy continuation approaches \cite{23,1} for the problem at hand, as solution paths merge and do not generate all solutions.

We therefore employ a shrink-or-split strategy, based on subdivision of polynomials in Bézier form (see e.g. \cite{8,28}). This has been shown to be highly suitable to find real roots (see e.g the work of Gaukel, Elber and Grandine, and Mourrain et al., Reuter et al. \cite{13,29,24,27}). The approach recursively delimits ever smaller parameter regions where roots can occur; and discards the remainder. Often, existence of a root can be proven; otherwise refinement to the prescribed tolerance allows discarding ambiguous regions that would be home to unstable roots. Crucially, the approach excels at finding real roots and provides guarantees on the location of roots also in the presence of moderate perturbations when approximating molecule parts by a collection of balls.

6 User Interface

Fig. 11 summarizes EASAL. EASAL accepts PDB data input or an existing atlas Fig. 11-left, allows for real-time intervention to direct the sampling process and for access to the atlas for different queries or views at any stage of the sampling process.

The views include the hierarchy of parametrized charts in several dimensions, display of active constraint graphs, of the lower dimensional boundaries of a given configuration space and its parametrized chart. The atlas view shows the atlas as it is being built, as a tree of nodes each representing a active constraint graph. Selecting a node to display its ancestors and descendants unclutters the view. A spring & repulsion algorithm yield a good layout of the initial and small atlas views, Clicking a node loads its data and centers on it if they are available and else triggers its exploration. A small display (upper right) shows the Euclidean realization of the selected node.

The atlas view has two dialogs. A specific active constraint graph can be searched in the atlas as in Fig. 12 or selected for sampling if it is not found. Once a active constraint graph is sampled, the parametrized chart view Fig. 13(a) shows green cubes (proportional to the step size chosen) where parameters do not resulting in collision. Clicking a cube displays, in the upper right corner, a configuration associated with the parameter value. EASAL can also display parameters resulting in irreconcilable intermolecular collisions or that are not realizable. For more than three dimensions, arrow-controlled sliders select 3D slices. Since interior points are easily occluded, the third dimension can also be switched to a slider. The left side of the view enumerates newly formed boundaries of the active constraint region and enumerates color-coded boundaries.

The configuration space view Fig. 13 (b) shows realizations with constraints and parameters displayed as lines. Valid realization flips are shown on the lower right side of the view. One option to view the valid parameters in the order of detection is via ‘video controls’ (bottom right) with reverse, pause, play and stop options.

A user can intervene in the sampling process, for example by proposing the constraint pairs of a
Figure 11: **EASAL overview.** The core Sampling and Realization algorithms (top, right) build up a database of explored regions of the configuration space represented as the atlas. A user can intervene or query and view the current state of the atlas.

Figure 12: Configuration creation dialog: *left:* data with index numbers used to select contact pairs. The dials rotate the view of the data set.

active constraint graph for sampling. The Sampling algorithm always first searches the atlas to make sure it only generates unexplored active constraint graphs. Other options are to refine the sampling in
a region, limit it, complete a particular dimension or stop the sampling of a particular active constraint region in the atlas.

The EASAL-implementation is object-oriented, uses OpenGL and the open-source widget FOX-toolkit [http://www.fox-toolkit.org/] for display to be portable across platforms. A key point of the design is to keep the memory profile low even when generating a large atlas. GUI and sampling algorithm use multiple threads.

7 Datasets and Computational experiments

In addition to numerous computational experiments on atlases for toy molecular data, we used EASAL to generate atlases of two alpha helices packing in transmembrane proteins, and dimeric assemblies of the AAV2 coat protein. Below is a brief sketch of these experiments meant only to illustrate proof of concept. Significant details are in [20] and [19].

7.1 Sampling transmembrane helix configuration space, MC comparision

We compare EASAL to Metropolis Monte Carlo Sampling MMCS [22] for generating configuration spaces of two alpha helices packing in transmembrane proteins with steric, membrane, volume, lipid,
interhelical axis angle and solvation energy based constraints. Fig. 14 shows the resulting EASAL atlas stratified according to active constraint regions. Fig. 15 shows the features of the configuration space for a randomly chosen active constraint region of the atlas.

Known challenges for MMCS include: (i) sampling outside the feasible region leading to many discarded samples; (ii) non-sampling of the configuration space; (iii) non stochastic: a high energy barrier can stop MMCS from sampling entire region of lowish energy beyond the barrier. EASAL solves these problems: (i) By convex parametrization of configuration space and efficiently detecting boundaries, EASAL does not sample infeasible regions. (ii) EASAL is aware if a region is sampled before or not thanks to stratification of configuration space. (iii) EASAL’s exhaustive sampling with higher refinement at the lower dimensions can find lowish energy configurations. Besides, EASAL can help MMCS to find high energy barriers. Fig. 16 shows such a low energy configuration (blue region) not covered by MMCS. Here we represented the 6 degrees of freedom of each configuration by a quaternion having 6 components. We project MMCS trajectory data [18] and the EASAL configurations onto the eigenbasis of the trajectory data and then plot pairs of eigenbasis vectors.

Figure 16: 2d projection of configuration spaces showing the proportion of sampling coverage EASAL (blue) over Monte Carlo (red).

### 7.2 Crucial constraints

In order to understand supramolecular and macromolecular self-assembly, such as viral self-assembly, it helps to isolate those intermolecular interactions that drive assembly of a collection of molecular units. The atlas generated in EASAL is accurate enough to highlight differences in the assembly configuration space that occur when an intermolecular interaction is dropped. This allows predicting minimal sets of geometric assembly constraints whose removal disrupts the assembly of viral shells. Fig. 17b shows the change in the assembly configuration space when two such interactions are dropped from pentameric interfaces of AAV2 (Adeno Associated Virus) shell. The interaction constraints were obtained from X-ray structure. See [17] for further substantive details.
Figure 17: (b) black nodes are affected by dropping two constraints in the constraint graph.

8 Possible Extensions

we can set probability distribution that specifies likelihood of each natural geometric region of the configuration space. solving both problems (i) and (ii) does not require the full sampling algorithm presented here. A significantly more efficient algorithm that generates the atlas (together with 1 witness configuration in each region) without sampling

Figure 18: EASAL configuration space views for two (a) and three (e) toy molecules with specific contacts active. Blue molecules are fixed, green molecules are flexible. Space view (c) includes only feasible configurations (green blocks), Space view (d) also includes colliding (red) configurations. (e) A third (yellow) molecule is attached to blue molecule and both fixed while green molecule kept flexible. Some portion of the configuration space is cut off because of the collision with the third molecule. Notice that some portion of green blocks of Fig. 18 is replaced by red blocks in the space view.
Extension of the algorithm to a small constant number of molecular units more than two has been shown to be sufficient for dealing with arbitrarily large assemblies, using a multi-scale approach that employs decomposition into subassemblies and analyzing assembly pathways [31, 25, 2].

On the other hand, there are new algorithmic challenges of extending to more than two molecules (see future work). The time complexity for sampling of configuration space increases exponentially with the number of molecules in case brute force sampling is used unless we find an intelligent way of sampling using existing information of configuration space with 2 molecules.

The average realization complexity increases since the proportion of graphs with partial 3-tree property decreases and realizing a graph having not partial 3-tree is much time consuming than the ones having partial 3-tree.

Fig. [19] shows nice configurations created by EASAL.

Figure 19: EASAL screenshot: 3 different configurations with constraint graph under it from i, j, k perspectives.

There are straightforward extensions to the algorithm section: (a) permit an already partially generated atlas to be input; in this case algorithm proceeds from one of the unfinished regions of the current stratification; (b) start from a specified bi-tether or a specified region of the current stratification; (c) change the traversal of the stratification from depth to breadth first, for any specified region of the current stratification; (d) choose to only traverse specified regions of the current stratification; (e) allows increased sampling refinement for specified regions (f) limits stratification to regions satisfying global assembly constraints (g) extends stratification to include regions defined by active global assembly constraints.

9 Conclusion

EASAL is a novel method for exploring and analyzing high dimensional assembly configuration spaces. It is grounded by state-of-the art configuration space theory, and has performance guarantees. It promises to be especially useful for elucidating processes driven by configurational entropy. An efficient algorithm for computing the entropy, given the atlas, would be very valuable.
# Appendix: Pseudocode of Algorithms

| Variable | Meaning |
|----------|---------|
| M | molecule part: collection of atom markers (typically spheres) |
| A | assemblyConstraints: see Section 3 |
| h | stepSize when sample is true. |
| τ | tolerance for testing equality to 0 |
| sample | boolean, true when full sampling is required; false if only active constraint subregions need to be identified and the region’s volume is approximated for entropy |
| G | activeConstraintGraph: system of constraints currently active (could be represented by a contactGraph; small intervals rather than exact values) |
| parameters | set of parameters of a convex or optimally realizable parametrization except for the initial bi-tether regions, regions are only added into the stratification if an initial feasible configuration (witness, Cartesian realization) is found. |
| witnessConfig | Cartesian realizations of the wellconstrained system of G together with parameter values – before checking non-active constraints (e.g. collision) |
| initialRealizations | Arguments of subroutines that have not been changed from the calling routine’s input are omitted as indicated by dots. |

ConvexChartBoundaries(G, newParameters)

Leverage convex parametrization theory from Section 3 to obtain volume, bounds and description of boundaries of convex configuration space of G parametrized by newParameters.

AlreadySeen(G)  
Check whether the G is a new region.

Realize(G, parameters, parameterValues)

Find Cartesian realizations using Theorem 3.7, algorithm 33 to optimize algebraic complexity and algorithm 13 to find of all roots.

## References

[1] E. Allgower and K. Georg. *Numerical Continuation Methods: An Introduction*, volume 13 of *Series in Computational Mathematics*. Springer-Verlag, Berlin, 1990.

[2] M. Bóna, M. Sitharam, and A. Vince. Enumerating tree orbits under permutation group action and application to macromolecular assembly pathways. *Bull. Math. Biology, special issue on Algebraic Biology*, 2010. to appear.

[3] John F. Canny. Computing roadmaps of general semi-algebraic sets. In *AAECC-9: Proceedings of the 9th International Symposium, on Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, pages 94–107, London, UK, 1991. Springer-Verlag.

[4] D.A. Case, T.E. Cheatham, T. Darden, H. Gohlke, Jr. R. Luo, K.M. Merz, A. Onufriev, C. Simmerling, B. Wang, and R. Woods. The amber biomolecular simulation programs. *J. Computat. Chem.*, 26:1668–1688, 2005.
Algorithm 1: StartAtlas.  

StartAtlas

input : biTethers, M, A, h, τ, sample
output: ( create Stratification )

for each biTether do  // generate its G
    G := (vertices, edges, distances or small distance intervals a of the biTether)
        plus at least 2 other vertices so that constraints and parameters ( in
        GetConvexParameters) add up to 5;
    newParameters := GetConvexParameters (G,parameters);
    boundaries := ConvexChartBoundaries (G,newParameters);
    Search in boundaries for witness in newParameters' values;
    initialRealizations := Realize (G, newParameters, newParameterValues);
    Realizations := NonactiveConstraintCheck (G, initialRealizations, M);
    if Realizations is empty then  // stratum is empty
        Stratification[G] := empty; next biTether;
    end
    ExploreSubAtlas (G, empty, witnessConfigc,...);  // create Stratif[G]
end

Algorithm 2: ExploreSubAtlas

ExploreSubAtlas

input : G, parameters, witnessConfig, M, A, h, τ, sample
output: ( Stratification[G ] )

if G wellconstrained then  // part of zero-dim stratum
    InitialRealizations := Realize (G, empty, empty);
    Realizations := NonactiveConstraintCheck (G, initialRealizations, M);
    Stratification[G] := Realizations; return;
end

// For regions in higher-dimensional strata;
if G is not partial 3-tree then
    non3Explore(G, parameters, witnessConfig, ...); return ;
end

newParameters := GetConvexParameters (G,parameters)b;
boundaries := ConvexChartBoundaries (G,newParameters)b;
if sample then
    SampleExplore (G, newParameters, boundaries ...);
else
    SearchExplore (G, newParameters, boundaries ...);
end

Algorithm 2: ExploreSubAtlas  b if not already computed in StartAtlas.
**SampleExplore**

**input**: parameters, boundaries $G$, witnessConfig, $M$, $A$, $\tau$

**output**: \(\langle \text{Stratification}[G] \rangle\)

Starting from witnessConfig, Fig. [20]

for each step in boundaries do
  update parameters, newParameterValues;
  initialRealizations := Realize \((G, \text{parameters}, \text{newParameterValues})\);
  Realizations := NonactiveConstraintCheck \((G, \text{initialRealizations}, M)\);
  if Realizations is empty\(^1\) then
    find newWitness\(^2\) (and graph newG) by binary search with tolerance $\tau$;
    if not AlreadySeen(newG) then
      Stratification[newG] := ExploreSubAtlas(newG, newWitness, \ldots);
    end
    add newWitness, Realizations to region in Stratification[G];
    skip to next feasible\(^3\)
  end
end

**Algorithm 3**: SampleExplore: exhaustively explore boundaries with step size $h$.\(^1\) we are near a potentially interesting newly active constraint, and a new region of the stratification in a stratum of smaller dimension.\(^2\) witness a new region of the stratification defined by an active constraint graph newG and distance/angle (interval) constraint.\(^3\) indicated by child node, see explore.fig

**SearchExplore** is identical except that the for-loop is replaced by `while` search within boundaries returns new active constraint (with parameters, values) do.

**non3Explore**

**input**: $G$, parameters, witnessConfig, $M$, $A$

**output**: \(\langle \text{Stratification}[G] \rangle\)

add witnessConfig to Stratification[G];

if $G$ wellconstrained then  // part of zero-dim stratum to be re-solved by Realize
  return;
end

compute newParameters, newParameterValues of G;
newConstraintList := find closest atom pairs in M (M is positioned by witnessConfig);
for each constraint in newConstraintList that is not in A do
  newG := add new constraint to $G$;
  if notAlreadySeen(newG) then
    non3Explore(newG, newParameters, witnessConfig, $M$, $A$);
  end
end

**Algorithm 4**: non3Explore, a recursive routine to explore regions defined by constraints that do not have a well-constrained 3-realizable completion.
GenerateConvexParameters

input : G, parameters
output: newParameters

// Leverage theory of convex parametrization from Section 3
if G is 3-realizable then // partial 3-tree
    get newParameters as maximal 3-realizable (3-tree) extension (retaining old parameters as much as possible and having new parameters represent pairs of vertices in G with large distance interval constraints);
end
if maximal 3-realizable extension is not well-constrained then
    get additional newParameters by optimal extension to well-constrainedness ;
end
return newParameters;

Algorithm 5: GenerateConvexParameters

NonactiveConstraintCheck

input : G, initialRealizations,M
output: Realizations

for r in initialRealizations do
    transform M according to r;
    if any constraints not in G have become active then // collision
        return ;
    else
        return empty; // Binary search uses violated constraints
    end
end

Algorithm 6: NonactiveConstraintCheck

Figure 20: Witness configuration with different parameters than parent node. (top) Parent node with 3 assemblyConstraints. (right) Child node with 4 assemblyConstraints.
[5] U. Chittamuru. *Efficient Iterative algorithm for bounding and sampling the Cayley configuration space of partial 2-trees in 3D*. M.S. Thesis University Of Florida, 2010.

[6] L. Dupont, D. Lazard, S. Lazard, and S. Petitjean. Near-optimal parameterization of the intersection of quadrics. In *COMPGEOM: Annual ACM Symposium on Computational Geometry*, 2003.

[7] I.Z. Emiris, E. Fritzilas, and D. Manocha. Algebraic algorithms for conformational analysis and docking. *Intern. J. Quantum Chemistry, Special Issue on Symbolic algebra in computational chemistry*, 2006.

[8] Gerald Farin. *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide*. Academic Press, 1988.

[9] MEERA fix citation! bib:survey.

[10] MEERA fix citation! likely sitharam:2010:ops.

[11] MEERA fix citation! likely sitharam:2010:rcc.

[12] MEERA fix citation! rojas.

[13] J. Gaukel. *Effiziente Losung polynomialer und nichtpolynomialer Gleichungssysteme mit Hilfe von Subdivisionsalgorithmen*. PhD thesis, University of Stuttgart, 2003.

[14] Jack E. Graver, Brigitte Servatius, and Herman Servatius. *Combinatorial Rigidity*. Graduate Studies in Math., AMS, 1993.

[15] Mike Thorpe's group at ASU. *FRODA: Framework rigidity optimized dynamics algorithm*. http://flexweb.asu.edu/software/, 2006.

[16] M. Karplus and J.N. Kushick. Method for estimating the configurational entropy of macromolecules. *Macromolecules*, 14(2):325–332, 1981.

[17] M. McKenna Lab. AAV virus assembly data.

[18] Maria Kurnikova Lab. Mmcs trajectory data.

[19] Sitharam M. and Ozkan. Sampling transmembrane helix configuration space using easal and mc comparison.

[20] Sitharam M., Wu, and Ozkan. Finding crucial constraints of viral self-assembly using easal.

[21] Anthony man Cho So and Yinyu Ye. A semidefinite programming approach to tensegrity theory and realizability of graphs. pages 766–775, 2006. Symposium on Discrete Algorithms.

[22] Nicholas Metropolis and S. Ulam. The monte carlo method. *Journal of the American Statistical Association*, 44(247):335–341, 1949.

[23] Alexander Morgan. *Solving Polynomial Systems Using Continuation for Engineering and Scientific Problems*. Prentice-Hall, 1987.

[24] B. Mourrain and J. P. Pavone. Subdivision methods for solving polynomial equations. *Journal of Symbolic Computation*, 44(3):292–306, March 2009.
[25] Bóna M. and Sitharam M. Influence of symmetry on probabilities of icosahedral viral assembly pathways. *Computational and Mathematical Methods in Medicine: Special issue on Mathematical Virology, Stockley and Twarock Eds*, 2008.

[26] M. Raghavan and B. Roth. Inverse kinematics of the general 6R manipulator and related linkages. *J. of Mech. Design, Trans ASME*, 115(3), 1993.

[27] Martin Reuter, Tarjei S. Mikkelsen, Evan C. Sherbrooke, Takashi Maekawa, and Nicholas M. Patrikalakis. Solving nonlinear polynomial systems in the barycentric Bernstein basis. *The Visual Computer*, 24(3):187–200, 2008.

[28] T. Sederberg and T. Nishita. Curve intersection using Bézier clipping. *Computer Aided Design*, 22(9):538–549, 1990.

[29] E. C. Sherbrooke and N. M. Patrikalakis. Computation of the solutions of nonlinear polynomial systems. *Computer Aided Geometric Design*, 10(5):379–405, October 1993.

[30] M. Sitharam. Characterizing well-formed systems of incidences for resolving collections of rigid bodies. *IJCGA*, 16(5-6):591–615, 2006.

[31] M. Sitharam and M. Agbandje-McKenna. Modeling virus assembly using geometric constraints and tensegrity: avoiding dynamics. *Journal of Computational Biology*, 13(6):1232–1265, 2006.

[32] M. Sitharam and H. Gao. Characterizing graphs with convex cayley configuration spaces. *Discrete and Computational Geometry*, 2010. to appear.

[33] M. Sitharam, J. Peters, and Yong Zhou. Optimized parametrization of systems of incidences between rigid bodies. *Journal of Symbolic Computation*, 45:481–498, 2010.

[34] Maria Belk (Sloughter) and Robert Connelly. Realizability of graphs. *Discrete Computational Geometry*, 37(2):125–137, 2007.

[35] Thom-Whitney. [http://en.wikipedia.org/wiki/Topologically_stratified_space](http://en.wikipedia.org/wiki/Topologically_stratified_space)

[36] W. Wang, B. Joe, and R. N. Goldman. Computing quadric surface intersections based on an analysis of plane cubic curves. *Graphical Models*, 64(6):335–367, 2002.