University of California
San Francisco

Integrative modeling with IMP
SEA complex
A case study with ambiguity and low structural coverage

Ilan E. Chemmama, Ignacia Echeverria, Ph.D., Seung Joong Kim, Ph.D.
Andrej Šali Laboratory
Workshop on Computational Biophysics
University of California, San Francisco
16 December 2016

Integrative modeling of the SEA complex

Gathering Data

Representing and Translating Data Into Spatial Restraints

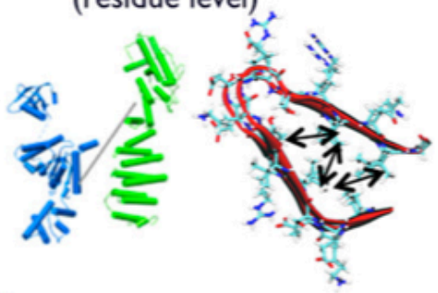
Sampling the Good Scoring Configurations

Analyzing and Assessing the ensemble

Experimental data

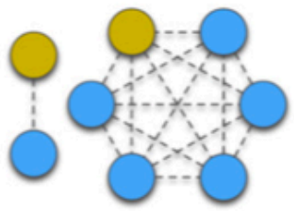
Residue-specific cross-linking
45 inter-molecular and
143 intra-molecular cross-links

Inter- and intra-molecular distance restraints (residue level)



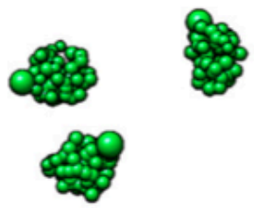
Protein and domain interactions
23 affinity purifications

Domain connectivity (Composites)



Stoichiometry
8 proteins

Number of subunits



X-ray crystallography
2 proteins / domains

Atomic structures

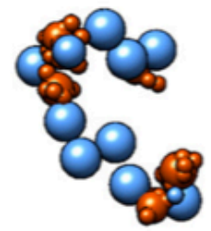
Statistical inference and physical principles

Comparative modeling
19 domains

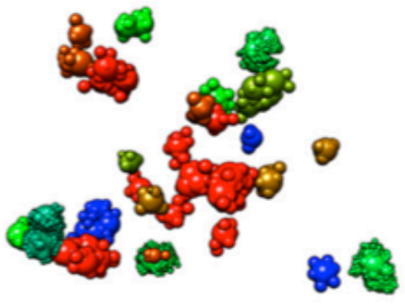
Fold models

Bioinformatics
8 proteins

Excluded Volume



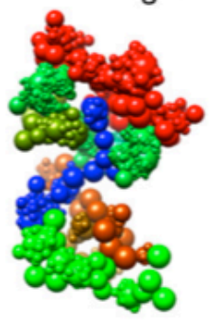
Random configurations



Initial sampling

(Cross-links and selected composite restrains)

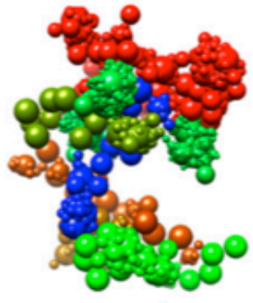
Connected configurations



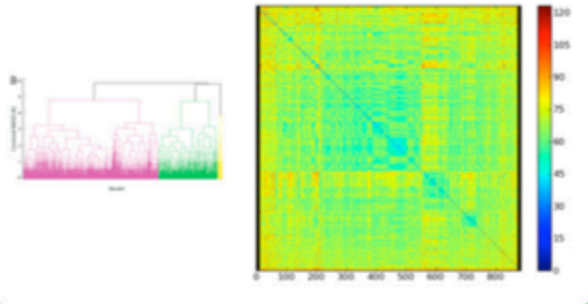
Refinement

(Cross-links and ALL composite restrains)

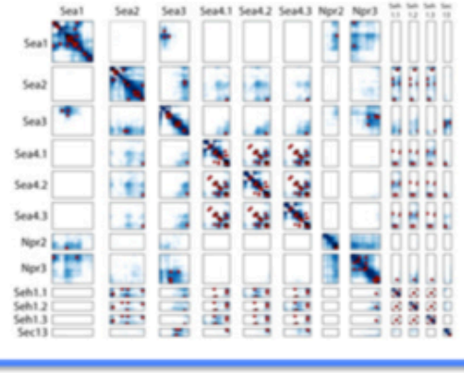
Final structures



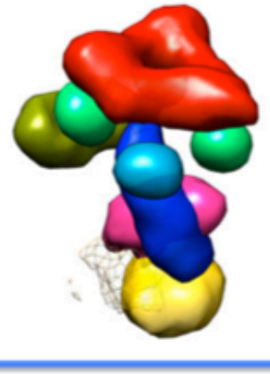
Model clustering



Protein and domain contact frequency



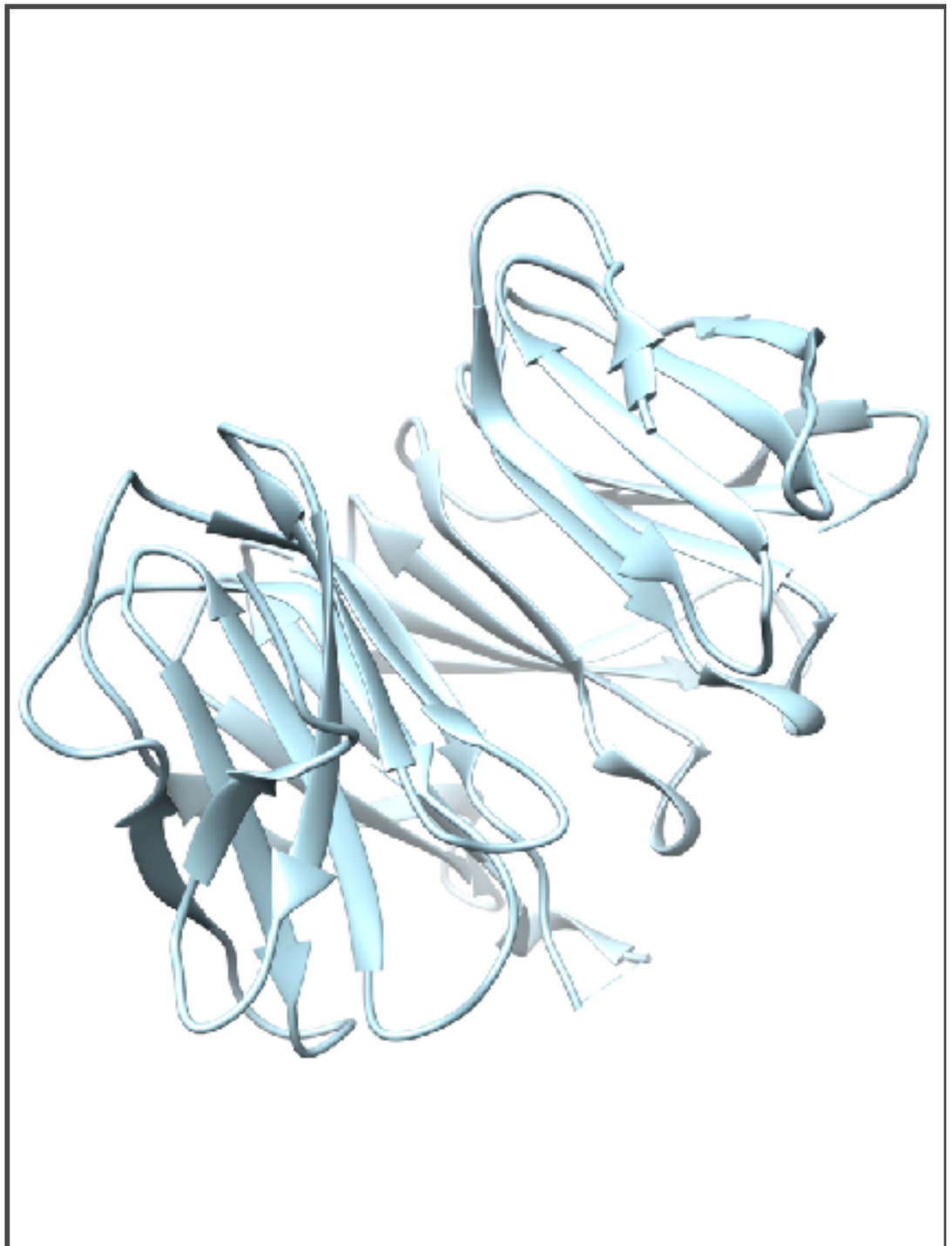
Protein and domain localization



Gathering Data, Representing the system and Translating them into spatial restraints

Prior structural knowledge: crystal structures

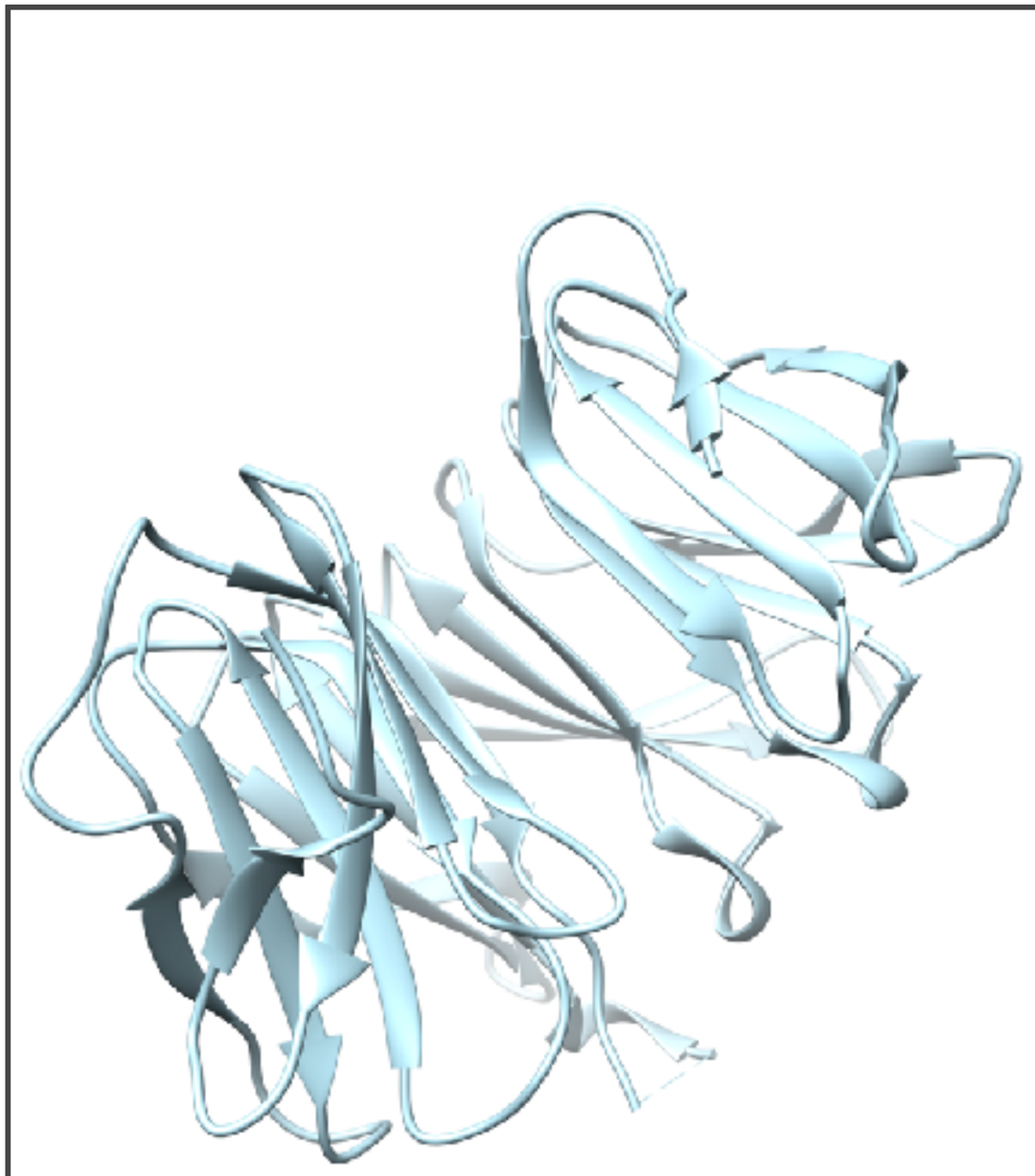
Prior structural knowledge: crystal structures



Sec13

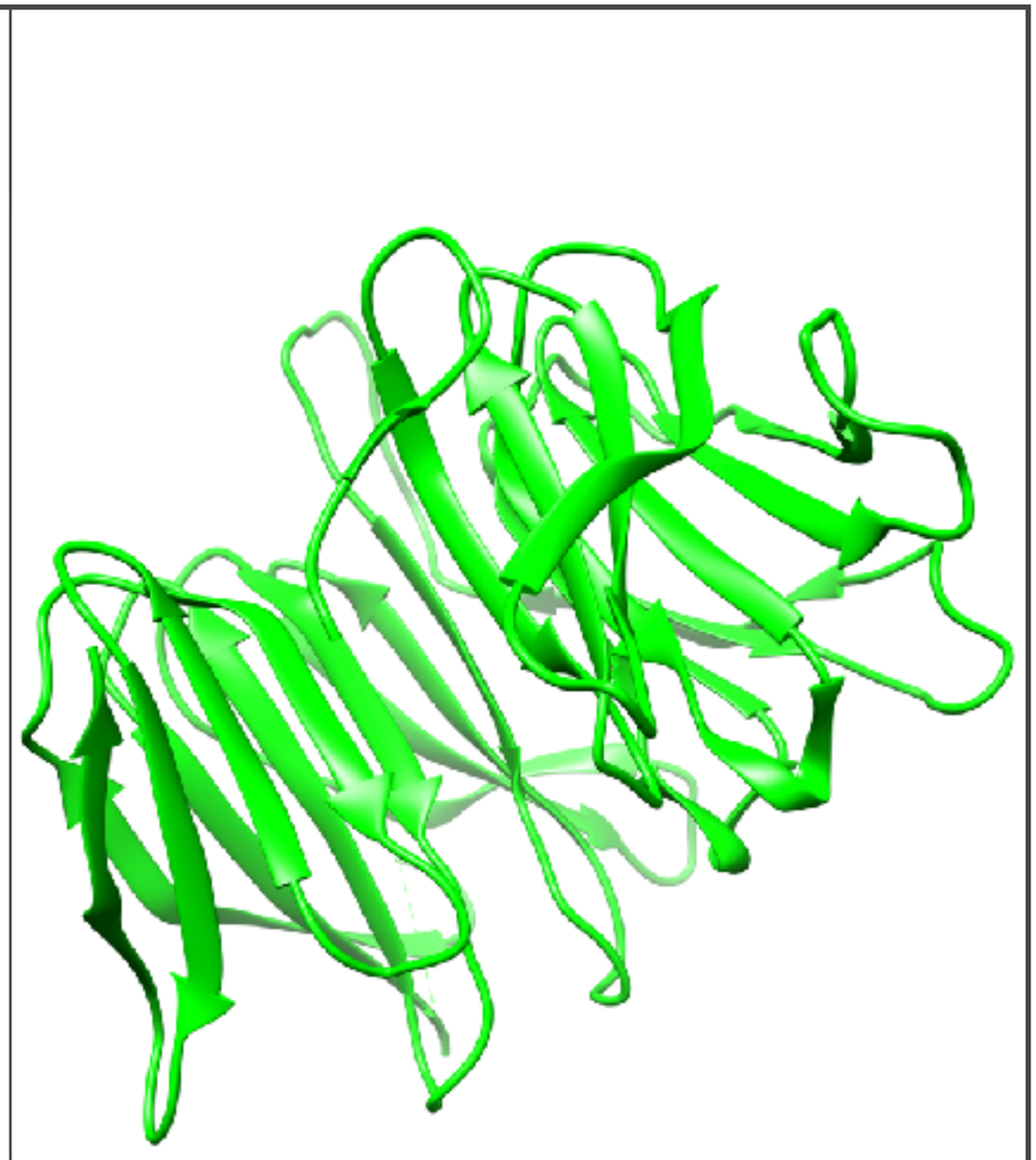
PDB code: 2PM7

Prior structural knowledge: crystal structures



Sec13

PDB code: 2PM7

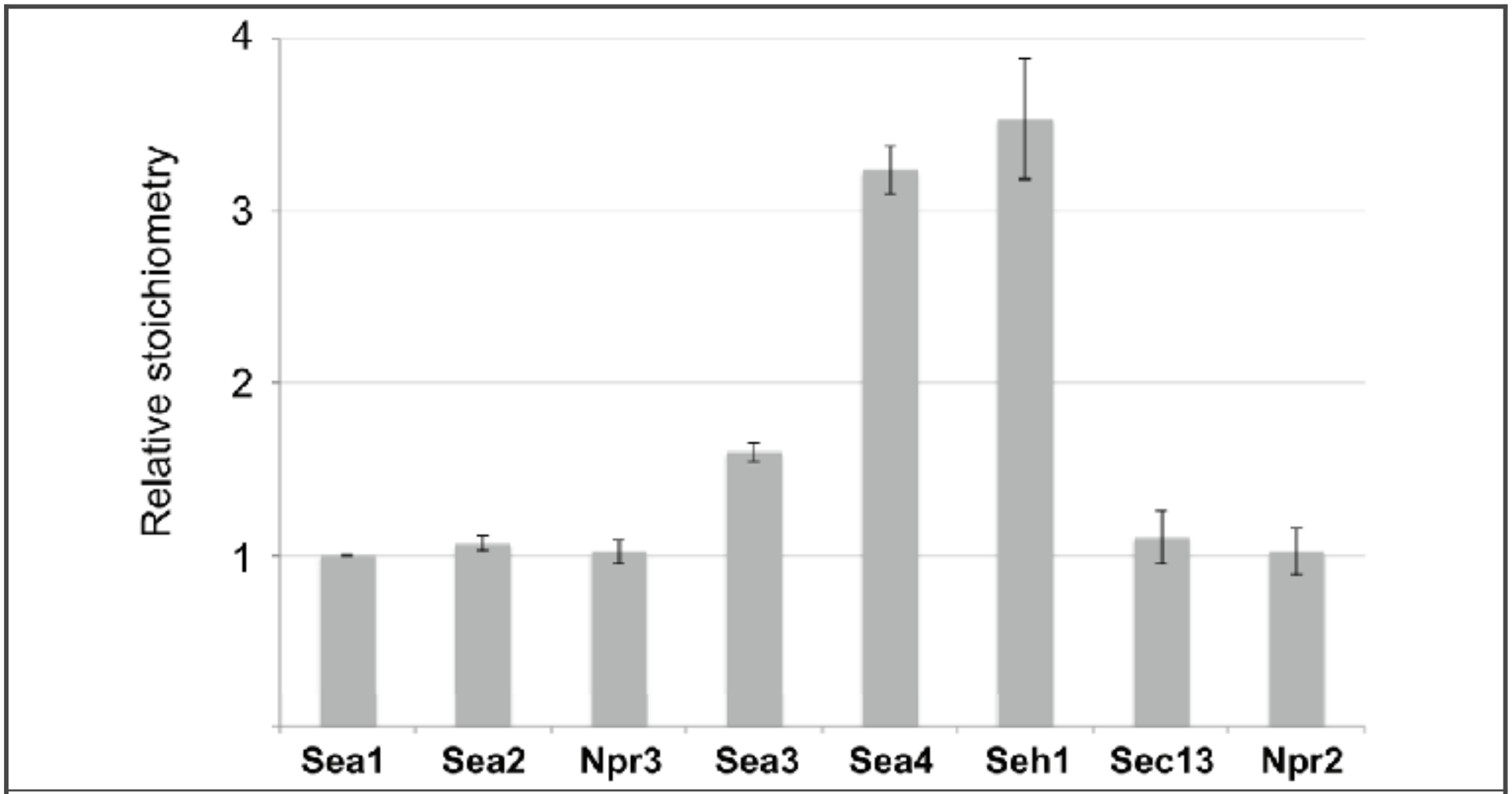


Seh1

PDB code: 3F3F

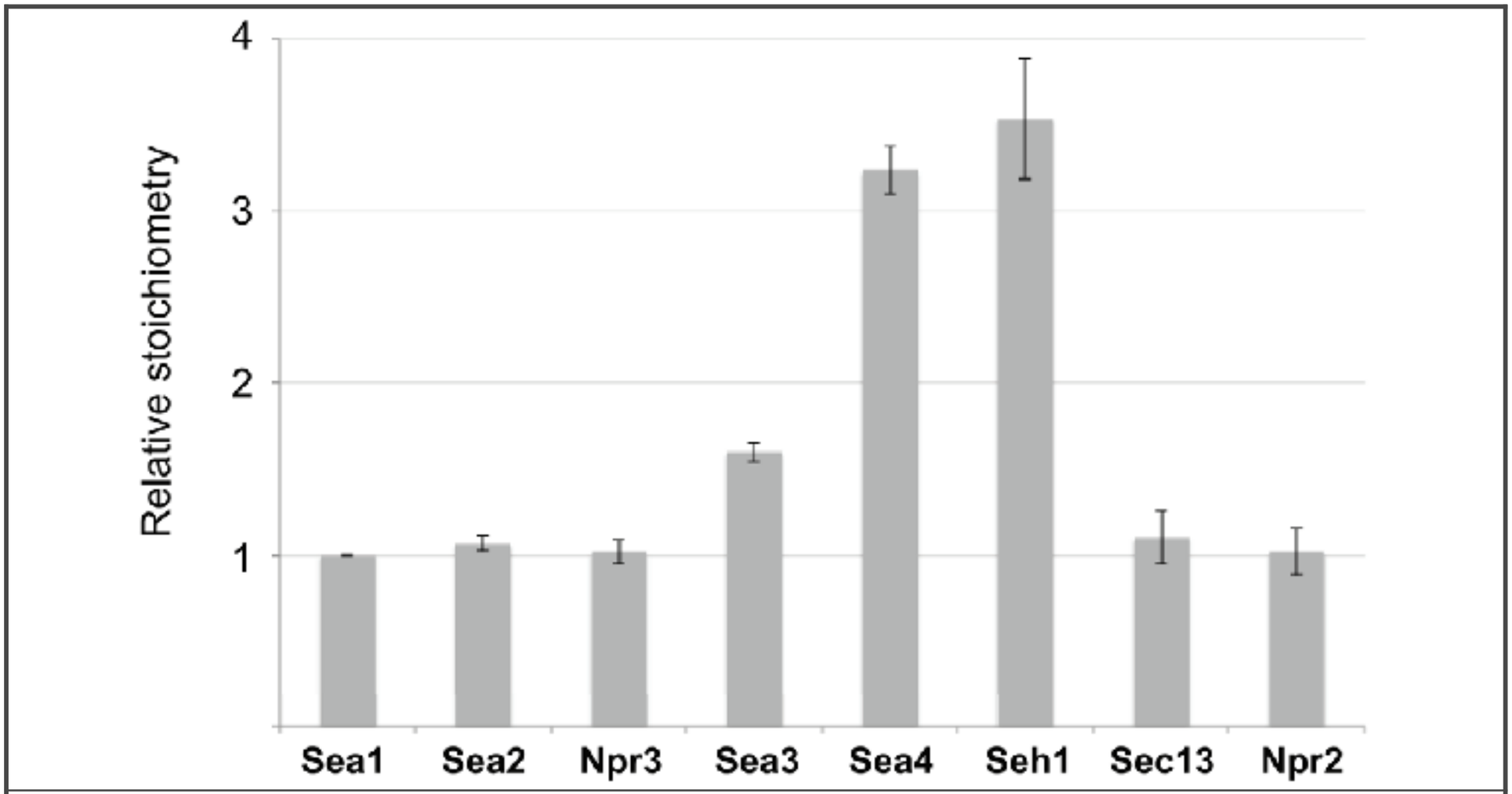
Stoichiometry

Stoichiometry



Analysis of the SEA complex relative stoichiometry by staining the resulting fractions of a Bis-Tris gel using SYPRO Ruby staining.

Stoichiometry



Analysis of the SEA complex relative stoichiometry by staining the resulting fractions of a Bis-Tris gel using SYPRO Ruby staining.

Npr2:Npr3:SEA1:SEA2:SEA3:SEA4:Sec13:Seh1 = 1:1:1:1:1:3:1:3

Representation in IMP 2.6.2:Topology

Representation in IMP 2.6.2:Topology

m = IMP.Model() topology = IMP.pmi.topology.TopologyReader(topology_file) domains = topology.component_list	Representation for the topology of the biomolecules is build using prior structural knowledge such as stoichiometry, crystal structures (Sec13, Seh1), comparative models built from homologs, secondary structure predictions,….

Representation in IMP 2.6.2:Topology

```
m = IMP.Model()

topology =
IMP.pmi.topology.TopologyReader(topology_file)
domains = topology.component_list
```

Representation for the topology of the biomolecules is build using prior structural knowledge such as stoichiometry, crystal structures (Sec13, Seh1), comparative models built from homologs, secondary structure predictions,….

component_name	domain_name	fasta_fn	fasta_id	pdb_fn	chain	residue_range	pdb_offset	bead_size	em_residues_per_gaussian
Npr2	Npr2_1	SeaComplex.fasta	Npr2		A	1, 8	0	10	0
Npr2	Npr2_2	SeaComplex.fasta	Npr2	Npr2_9-127.pdb	A	9,127	0	10	0
Npr2	Npr2_3	SeaComplex.fasta	Npr2		A	128,256	0	65	0
Npr2	Npr2_4	SeaComplex.fasta	Npr2	Npr2_257-327.pdb	A	257,327	0	10	0
Npr2	Npr2_501	SeaComplex.fasta	Npr2		A	328,355	0	28	0
Npr2	Npr2_502	SeaComplex.fasta	Npr2		A	356,430	0	75	0
Npr2	Npr2_503	SeaComplex.fasta	Npr2		A	431,493	0	63	0
Npr2	Npr2_504	SeaComplex.fasta	Npr2		A	494,562	0	69	0
Npr2	Npr2_6	SeaComplex.fasta	Npr2	Npr2_563-610.pdb	A	563,610	0	10	0
Npr2	Npr2_7	SeaComplex.fasta	Npr2		A	611,615	0	10	0
Seh1.1	Seh1.1_2	SeaComplex.fasta	Seh1	3F3F.pdb	A	1,248	0	10	0
Seh1.1	Seh1.1_3	SeaComplex.fasta	Seh1		A	249,287	0	39	0
Seh1.1	Seh1.1_4	SeaComplex.fasta	Seh1	3F3F.pdb	A	288,346	0	10	0
Seh1.1	Seh1.1_5	SeaComplex.fasta	Seh1		A	347,349	0	10	0
Seh1.2	Seh1.2_2	SeaComplex.fasta	Seh1	3F3F.pdb	A	1,248	0	10	0
Seh1.2	Seh1.2_3	SeaComplex.fasta	Seh1		A	249,287	0	39	0
Seh1.2	Seh1.2_4	SeaComplex.fasta	Seh1	3F3F.pdb	A	288,346	0	10	0
Seh1.2	Seh1.2_5	SeaComplex.fasta	Seh1		A	347,349	0	10	0
Seh1.3	Seh1.3_2	SeaComplex.fasta	Seh1	3F3F.pdb	A	1,248	0	39	0
Seh1.3	Seh1.3_3	SeaComplex.fasta	Seh1		A	249,287	0	10	0
Seh1.3	Seh1.3_4	SeaComplex.fasta	Seh1	3F3F.pdb	A	288,346	0	10	0
Seh1.3	Seh1.3_5	SeaComplex.fasta	Seh1		A	347,349	0	10	0
Sec13	Sec13_1	SeaComplex.fasta	Sec13		D	1,1	0	10	0
Sec13	Sec13_2	SeaComplex.fasta	Sec13	2PM7.pdb	D	2,158	0	10	0
Sec13	Sec13_3	SeaComplex.fasta	Sec13		D	159,165	0	10	0
Sec13	Sec13_4	SeaComplex.fasta	Sec13	2PM7.pdb	D	166,296	0	10	0
Sec13	Sec13_5	SeaComplex.fasta	Sec13		D	297,297	0	10	0

Representation and Movers

Representation and Movers

```
rigid_bodies=[
                                    [“Npr2_2"], ["Npr2_4"], ["Npr2_6"],
                                    ["Npr3_1"],
                                    ["Npr3_3", "Npr3_5"],
                                    ["Npr3_7"], ["Npr3_9"], [“Npr3_11"],
                                    ["Sec13_2", “Sec13_4”],
                                    [“SEA1_2”],
                                    ["SEA1_4", "SEA1_6", “SEA1_8”],
                                    [“SEA1_10"],
                                    ["SEA2_2", "SEA2_4", “SEA2_6”, "SEA2_8"],
                                    ["SEA2_10"],
                                    ["SEA3_2", "SEA3_4", "SEA3_6", "SEA3_8"],
                                    [“SEA3_10"], ["SEA3_12"],
                                    ["Seh1.1_2", "Seh1.1_4"],
["SEA4.1_2", "SEA4.1_4", "SEA4.1_6", “SEA4.1_8”, "SEA4.1_10"],
                                    ["SEA4.1_12", "SEA4.1_14"],
                                    ["SEA4.1_16", "SEA4.1_18"]
]
```

Define the movers by defining the list of rigid body. A rigid body is a set of particle beads that will move together during the sampling stages.

Thus, a set of particle beads within a rigid body will contribute only 6 degrees of freedom to the sampling all together

=> huge speed up in computing time

Representation and Movers

```
rigid_bodies=[
                                        [“Npr2_2"], ["Npr2_4"], ["Npr2_6"],
                                        ["Npr3_1"],
                                        ["Npr3_3", "Npr3_5"],
                                        ["Npr3_7"], ["Npr3_9"], [“Npr3_11"],
                                        ["Sec13_2", “Sec13_4”],
                                        [“SEA1_2”],
                                        ["SEA1_4", "SEA1_6", “SEA1_8”],
                                        [“SEA1_10"],
                                        ["SEA2_2", "SEA2_4", “SEA2_6”, "SEA2_8"],
                                        ["SEA2_10"],
                                        ["SEA3_2", "SEA3_4", "SEA3_6", "SEA3_8"],
                                        [“SEA3_10"], ["SEA3_12"],
                                        ["Seh1.1_2", "Seh1.1_4"],
["SEA4.1_2", "SEA4.1_4", "SEA4.1_6", “SEA4.1_8”, "SEA4.1_10"],
                                        ["SEA4.1_12", "SEA4.1_14"],
                                        ["SEA4.1_16", "SEA4.1_18"]
]
```

Define the movers by defining the list of rigid body. A rigid body is a set of particle beads that will move together during the sampling stages.

Thus, a set of particle beads within a rigid body will contribute only 6 degrees of freedom to the sampling all together

=> huge speed up in computing time

```
rb_max_trans = 2.00
rb_max_rot = 0.04
bead_max_trans = 3.00
representation.set_rigid_bodies_max_rot(rb_max_rot)
representation.set_floppy_bodies_max_trans(bead_max_trans)
representation.set_rigid_bodies_max_trans(rb_max_trans)
```

Define how big the moves are during samples.

Building the system and define aesthetics

Building the system and define aesthetics

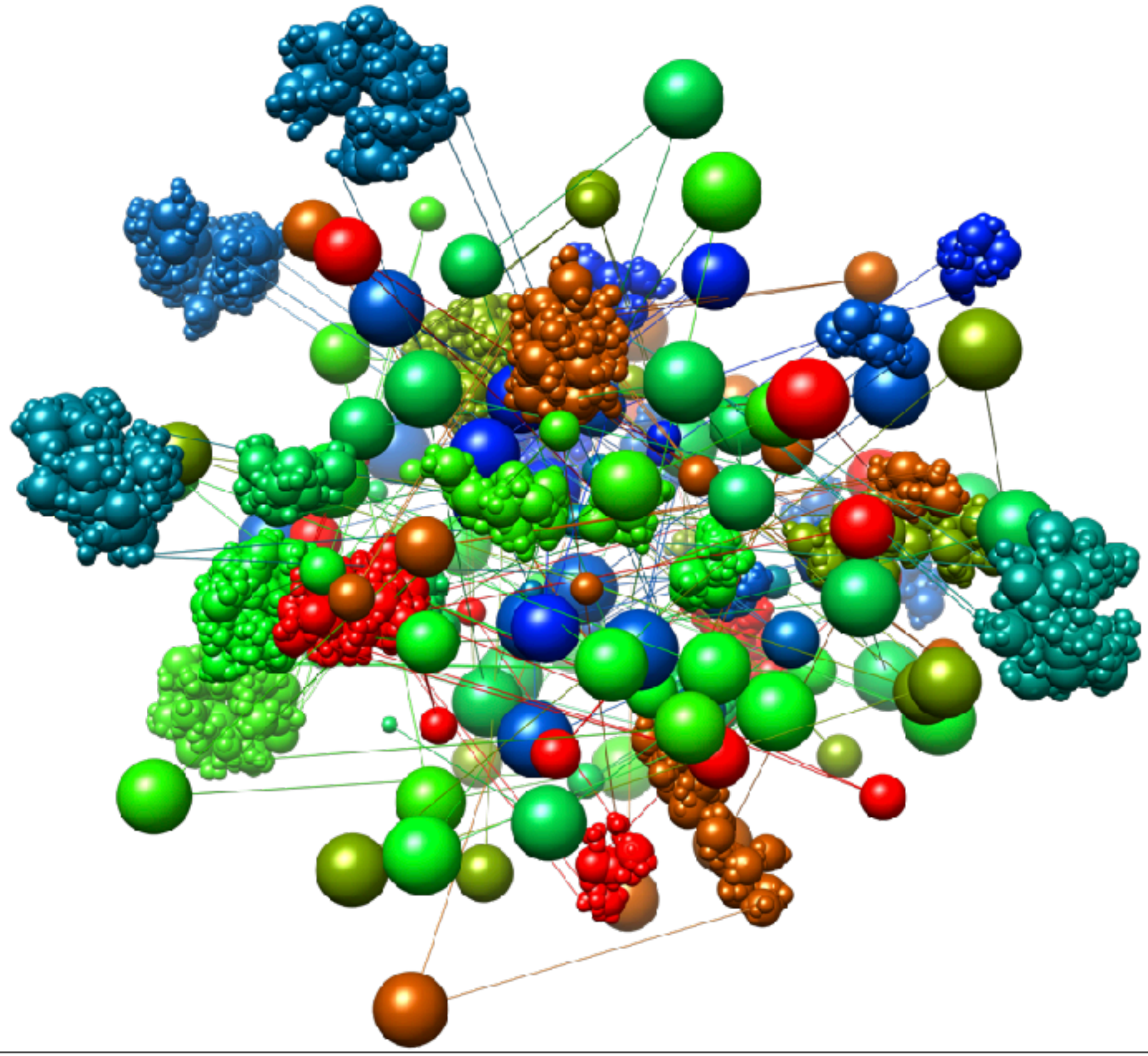
bm = IMP.pmi.macros.BuildModel(m, component_topologies=domains, list_of_rigid_bodies=rigid_bodies) representation = bm.get_representation()	Here, we build our representation, needed to start restraining part of our system

Building the system and define aesthetics

bm = IMP.pmi.macros.BuildModel(m, component_topologies=domains, list_of_rigid_bodies=rigid_bodies) representation = bm.get_representation()	Here, we build our representation, needed to start restraining part of our system
for nc,component **in** enumerate(domains): name = component.name sel = IMP.atom.Selection(representation.prot,molecule=name) ps = sel.get_selected_particles() clr = IMP.display.get_rgb_color(float(nc)/len(domains)) **for** p **in** ps: **if not** IMP.display.Colored.get_is_setup(p): IMP.display.Colored.setup_particle(p,clr) **else**: IMP.display.Colored(p).set_color(clr)	A little of aesthetic for the output: We select each one of the proteins and assign a color (not essential set-up to have good-looking output.

Example of a not-optimized conformation

Example of a not-optimized conformation



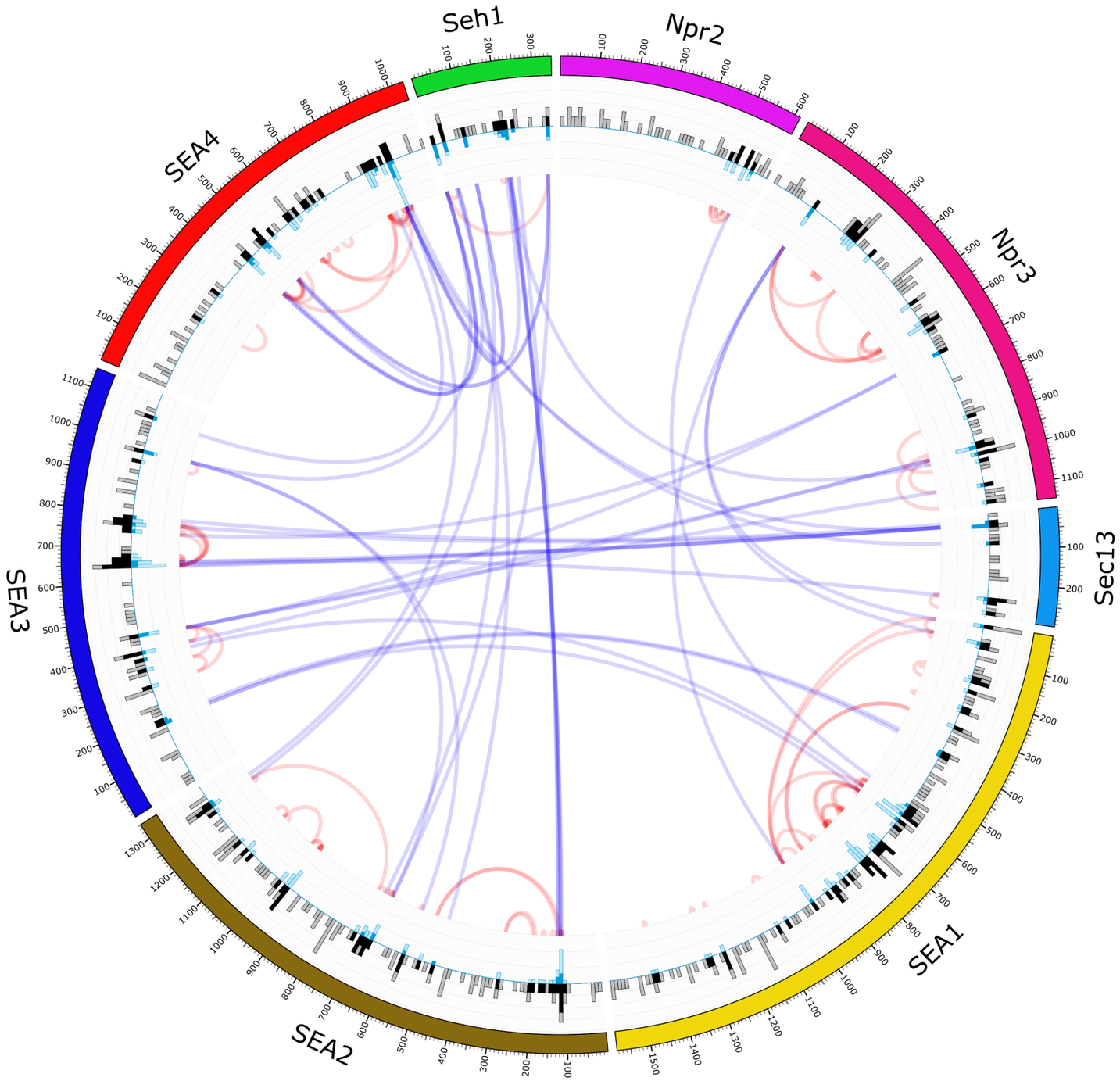
Initial shuffled configuration of the SEA complex.

Example of a physics-based restraint

```
ev =
IMP.pmi.restraints.stereochemistry.ExcludedVolumeSphere(
                                                representation,
                                                 resolution=10)
ev.add_to_model()
outputobjects.append(ev)
```

Excluded volume on beads with size closest to 10. This speeds up the excluded volume computation rather to compute it for individual residue bead

Residue-specific cross-links



Residue-specific cross-links used for the integrative modeling of the SEA complex. 45 inter-xls (**blue**) and 143 intra-xls (**red**) were obtained (188 total).

The **black** histogram represents the density of cross-links in the region.

Encoding the XL restraint

Encoding the XL restraint

prot1	res1	prot2	res2	id
Npr3	684	SEA3	743	1
SEA1	392	SEA3	189	2
SEA1	51	Npr3	132	3
SEA1	562	Npr2	562	4

```
kw = IMP.pmi.io.crosslink.CrossLinkDataBaseKeywordsConverter()
kw.set_unique_id_key("id")
kw.set_protein1_key("prot1")
kw.set_protein2_key("prot2")
kw.set_residue1_key("res1")
kw.set_residue2_key("res2")
kw.set_id_score_key(None)
xldb = IMP.pmi.io.crosslink.CrossLinkDataBase(kw)
xldb.create_set_from_file(datadirectory+'xlinks_SEA.csv')
```

Encoding the XL restraint

prot1	res1	prot2	res2	id
Npr3	684	SEA3	743	1
SEA1	392	SEA3	189	2
SEA1	51	Npr3	132	3
SEA1	562	Npr2	562	4

```
kw = IMP.pmi.io.crosslink.CrossLinkDataBaseKeywordsConverter()
kw.set_unique_id_key("id")
kw.set_protein1_key("prot1")
kw.set_protein2_key("prot2")
kw.set_residue1_key("res1")
kw.set_residue2_key("res2")
kw.set_id_score_key(None)
xldb = IMP.pmi.io.crosslink.CrossLinkDataBase(kw)
xldb.create_set_from_file(datadirectory+'xlinks_SEA.csv')
```

```
xl1 = IMP.pmi.restraints.crosslinking.CrossLinkingMassSpectrometryRestraint(
                                                    representation=representation,
                                                    CrossLinkDataBase=xldb,
                                                    length=21,
                                                    label="SeaComplex",
                                                    resolution=1.0,
                                                    slope=0.02)
```

This calls the IMP.pmi function for using XLs as spatial restraints using a Bayesian formalism of XL treatment.

Encoding the XL restraint

prot1	res1	prot2	res2	id
Npr3	684	SEA3	743	1
SEA1	392	SEA3	189	2
SEA1	51	Npr3	132	3
SEA1	562	Npr2	562	4

```
kw = IMP.pmi.io.crosslink.CrossLinkDataBaseKeywordsConverter()
kw.set_unique_id_key("id")
kw.set_protein1_key("prot1")
kw.set_protein2_key("prot2")
kw.set_residue1_key("res1")
kw.set_residue2_key("res2")
kw.set_id_score_key(None)
xldb = IMP.pmi.io.crosslink.CrossLinkDataBase(kw)
xldb.create_set_from_file(datadirectory+'xlinks_SEA.csv')
```

```
xl1 = IMP.pmi.restraints.crosslinking.CrossLinkingMassSpectrometryRestraint(
                                                    representation=representation,
                                                    CrossLinkDataBase=xldb,
                                                    length=21,
                                                    label="SeaComplex",
                                                    resolution=1.0,
                                                    slope=0.02)
```

This calls the IMP.pmi function for using XLs as spatial restraints using a Bayesian formalism of XL treatment.

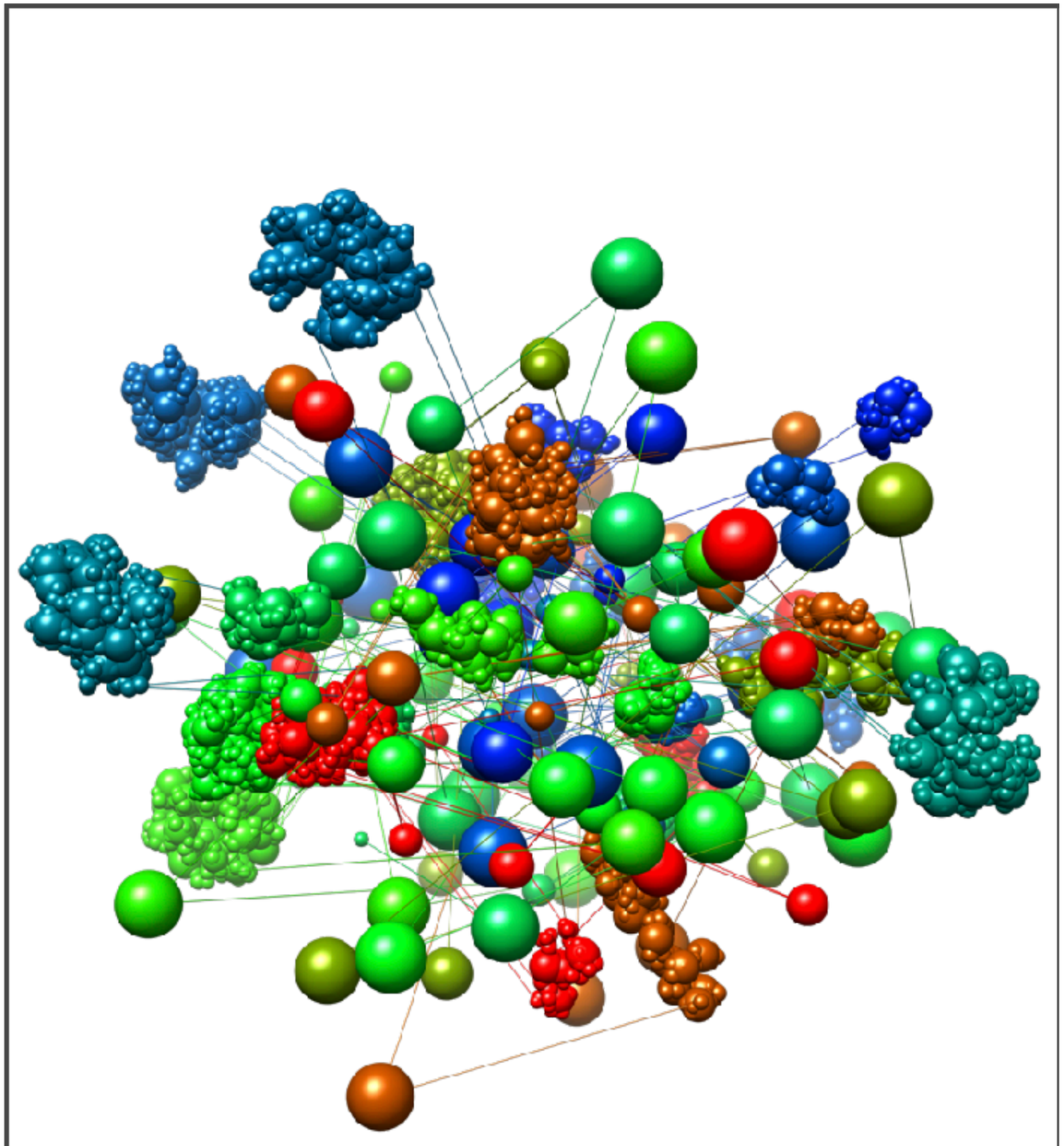
```
xl1.rs.set_weight(25.0)
xl1.add_to_model()

sampleobjects.append(xl1)
outputobjects.append(xl1)
```

Set the weight of the restraint relative to other restraint and add it to the model

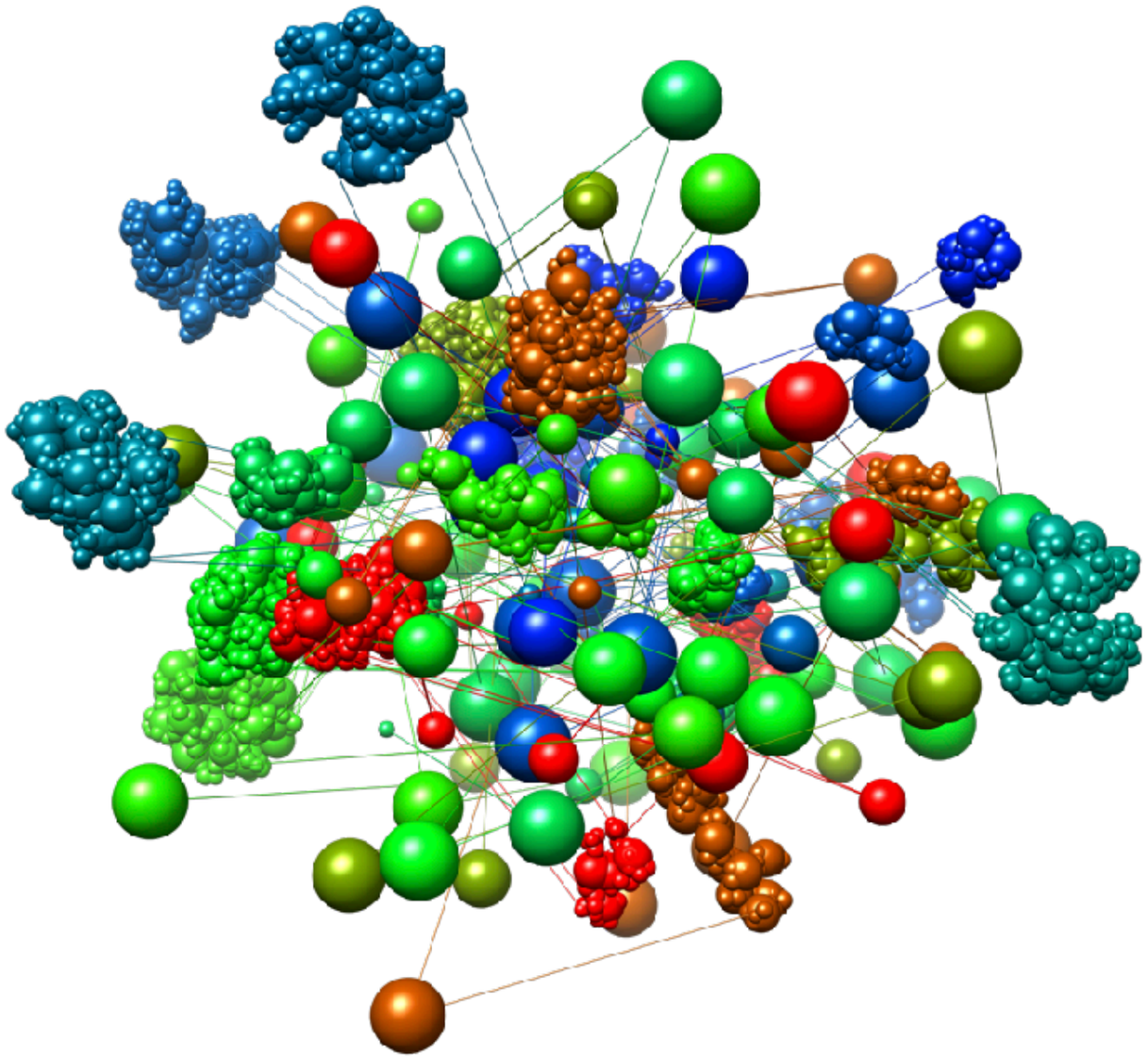
Monte Carlo Gibbs Sampling with Replica Exchange

Monte Carlo Gibbs Sampling with Replica Exchange

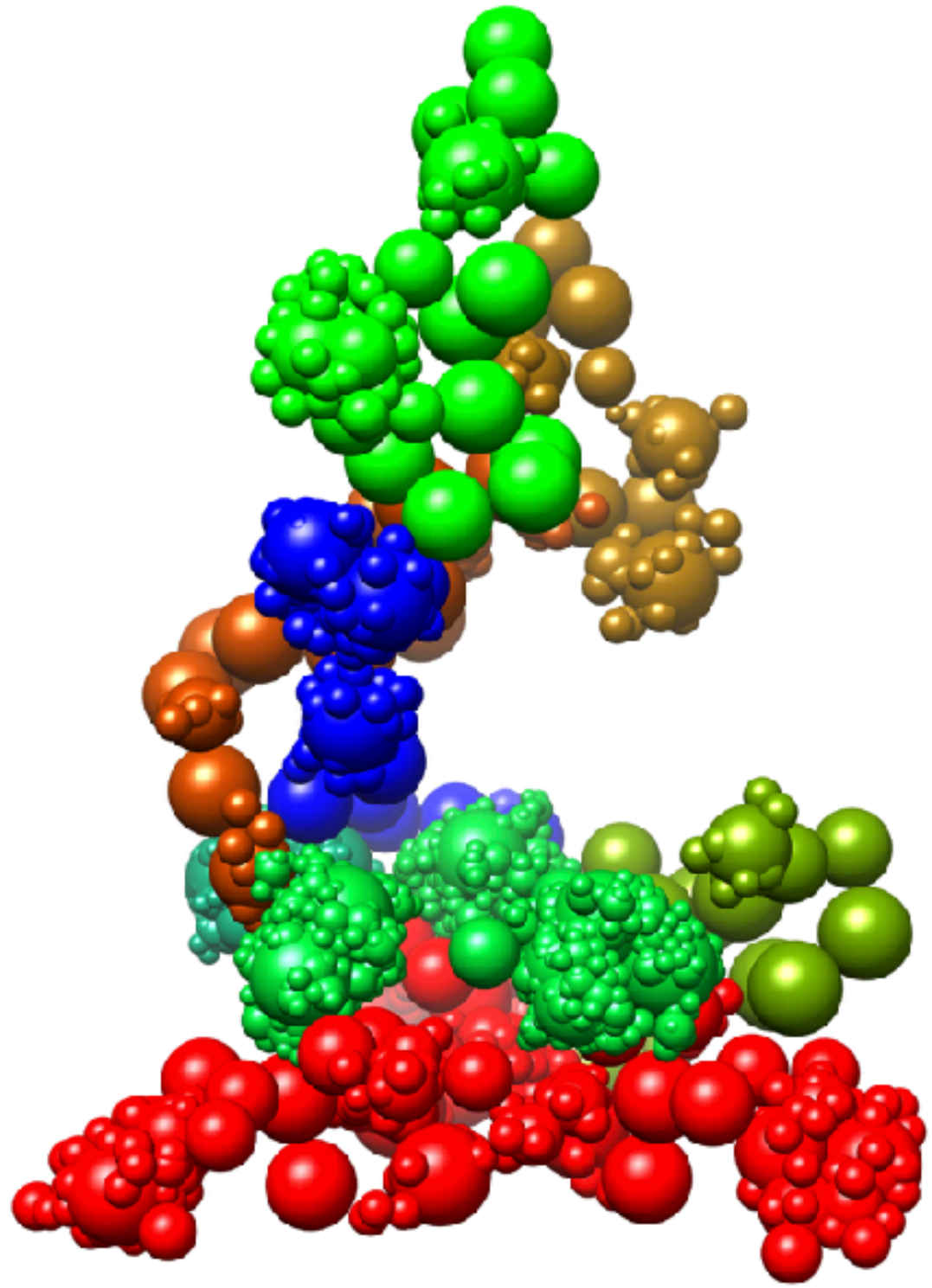


Initial shuffled configuration of the SEA complex.

Monte Carlo Gibbs Sampling with Replica Exchange



Initial shuffled configuration of the SEA complex.



Good-scoring model after sampling

Calling the Sampler!

Calling the Sampler!

representation.shuffle_configuration(50)	For each of the independent replicate trajectories, we start from a different “shuffled” configuration to ensure uncorrelated, independent trajectories.

Calling the Sampler!

representation.shuffle_configuration(50)	For each of the independent replicate trajectories, we start from a different "shuffled" configuration to ensure uncorrelated, independent trajectories.
mc1=IMP.pmi.macros.ReplicaExchange0(m, representation, monte_carlo_sample_objects=sampleobjects, output_objects=outputobjects, crosslink_restraints=[xl1,], monte_carlo_temperature=1.0, simulated_annealing=**True**, simulated_annealing_minimum_temperature=1.0, simulated_annealing_maximum_temperature=2.5, simulated_annealing_minimum_temperature_nframes=200, simulated_annealing_maximum_temperature_nframes=20, replica_exchange_minimum_temperature=1.0, replica_exchange_maximum_temperature=2.5, number_of_best_scoring_models=1, monte_carlo_steps=num_mc_steps, number_of_frames=5000, global_output_directory="output")	Define the sampling parameters for Monte Carlo Gibbs Sampling with Replica Exchange. If run with mpi, automatically setup replica exchange by evenly spaced independent replicas between replica_exchnage_minimum_temperature and replica_exchange_maximum_temperature for number_of_frames in the directory "./output"

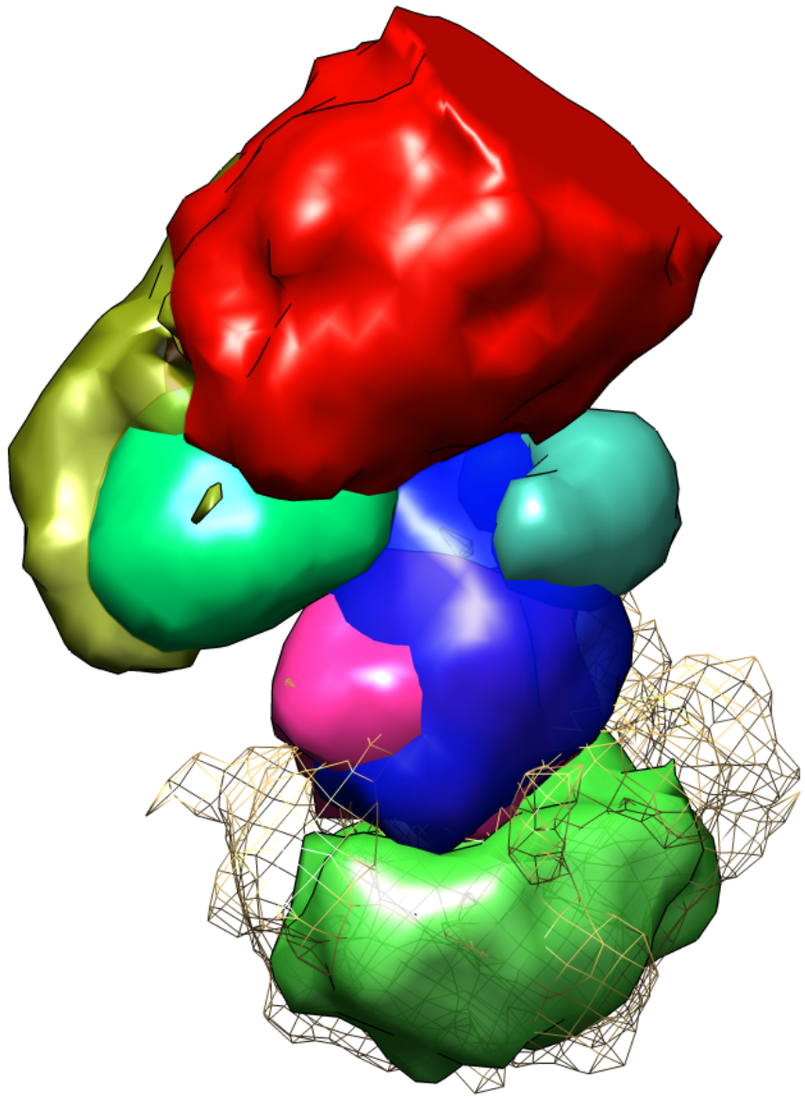
Calling the Sampler!

representation.shuffle_configuration(50)	For each of the independent replicate trajectories, we start from a different "shuffled" configuration to ensure uncorrelated, independent trajectories.
mc1=IMP.pmi.macros.ReplicaExchange0(m, representation, monte_carlo_sample_objects=sampleobjects, output_objects=outputobjects, crosslink_restraints=[xl1,], monte_carlo_temperature=1.0, simulated_annealing=**True**, simulated_annealing_minimum_temperature=1.0, simulated_annealing_maximum_temperature=2.5, simulated_annealing_minimum_temperature_nframes=200, simulated_annealing_maximum_temperature_nframes=20, replica_exchange_minimum_temperature=1.0, replica_exchange_maximum_temperature=2.5, number_of_best_scoring_models=1, monte_carlo_steps=num_mc_steps, number_of_frames=5000, global_output_directory="output")	Define the sampling parameters for Monte Carlo Gibbs Sampling with Replica Exchange. If run with mpi, automatically setup replica exchange by evenly spaced independent replicas between replica_exchnage_minimum_temperature and replica_exchange_maximum_temperature for number_of_frames in the directory "./output"
mc1.execute_macro()	Start the sampling by executing the sampler.

Before running a few frames… Spoilers

Adding symmetry

Adding symmetry

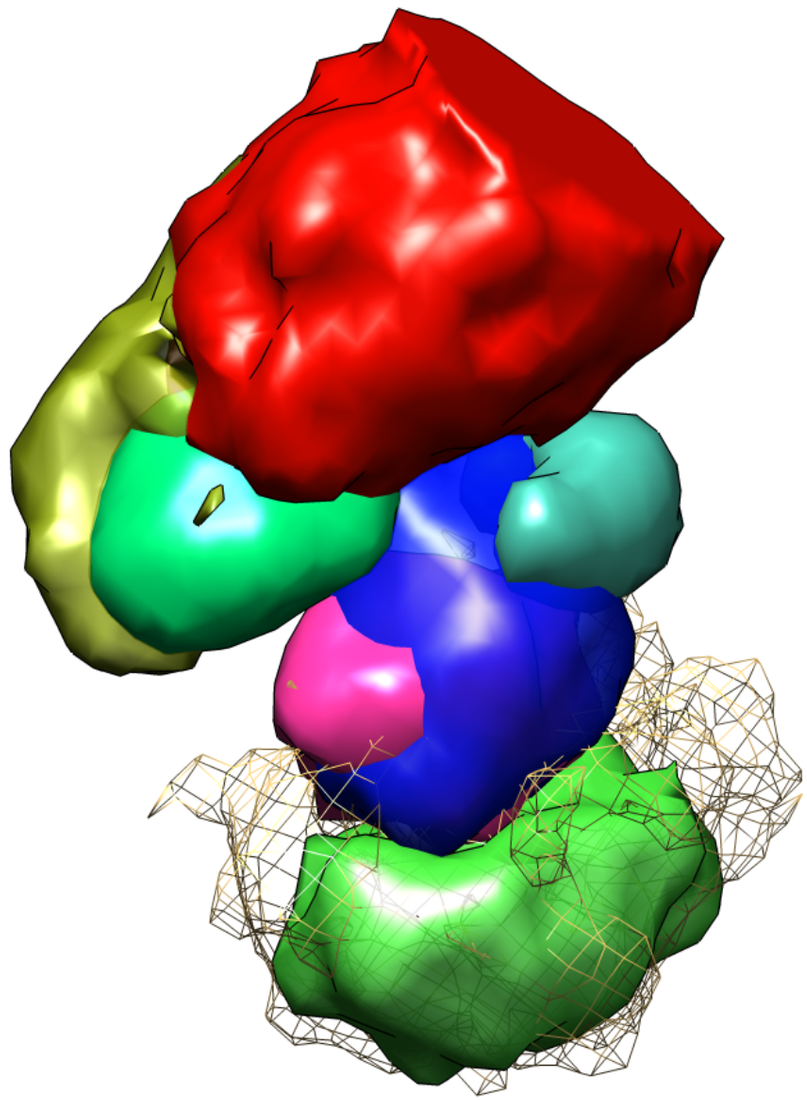


Resulting localization probability density for 1:3 stoichiometry but without symmetry.

Emergence of 3-fold symmetry (C3) for SEA4 and Seh1.

Constraint the system to have symmetry.

Adding symmetry



Resulting localization probability density for 1:3 stoichiometry but without symmetry.

Emergence of 3-fold symmetry (C3) for SEA4 and Seh1.

Constraint the system to have symmetry.

```
create_rotational_symmetry2(representation, "SEA4.1", ["SEA4.2","SEA4.3"])
create_rotational_symmetry2(representation, "Seh1.1", ["Seh1.2","Seh1.3"])
```

Set the weight of the restraint relative to other restraint and add it to the model

Encoding a symmetry constraint in IMP

```
def create_rotational_symmetry2(
                                                    rps,
                                                    maincopy,
                                                    copies,
                                    rotational_axis=IMP.algebra.Vector3D(0, 0, 1.0),
                                                    nSymmetry=None,
                                                    skip_gaussian_in_clones=False):

    from math import pi
    rps.representation_is_modified = True
    ncopies = len(copies) + 1
    main_hiers = IMP.atom.get_leaves(rps.hier_dict[maincopy])

    for k in range(len(copies)):
            if (nSymmetry is None):
                    rotation_angle = 2.0 * pi / float(ncopies) * float(k + 1)
            else:
                    if ( k % 2 == 0 ):
                            rotation_angle = 2.0 * pi / float(nSymmetry) * float((k + 2) / 2)
                    else:
                            rotation_angle = -2.0 * pi / float(nSymmetry) * float((k + 1) / 2)
            rotation3D = IMP.algebra.get_rotation_about_axis(rotational_axis, rotation_angle)
            sm = IMP.core.TransformationSymmetry(rotation3D)
            clone_hiers = IMP.atom.get_leaves(rps.hier_dict[copies[k]])

            lc = IMP.container.ListSingletonContainer(rps.m)
            for n, p in enumerate(main_hiers):
                if (skip_gaussian_in_clones):
                        if (IMP.core.Gaussian.get_is_setup(p)) \
                                and not (IMP.atom.Fragment.get_is_setup(p) or \
                                                IMP.atom.Residue.get_is_setup(p)):
                                continue
                pc = clone_hiers[n]
                IMP.core.Reference.setup_particle(pc.get_particle(), p.get_particle())
                lc.add(pc.get_particle().get_index())

                c = IMP.container.SingletonsConstraint(sm, None, lc)
                rps.m.add_score_state(c)
                print("Completed setting " + str(maincopy) + \
                                " as a reference for " + str(copies[k]) + \
                                " by rotating it in " + str(rotation_angle / 2.0 / pi * 360) + \
                                " degree around the " + str(rotational_axis) + " axis.")
            rps.m.update()
```

Create a symmetry constraint, given the main copy and its clone. The clones will have congruent configuration, transformed by **rotation_angle** and **rotational_axis**.

Let it run, let it run, let it run…

Two examples trajectories are provided as examples.

Analyzing the results

Best scoring models from all sampling runs

Split into two (or more) independent samples

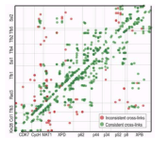
1. Sampling Convergence

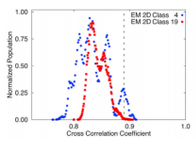
- Clustering
- Precision
- Visual Analysis
- χ^2 test

Localization density

Pass

2. Analyze fit to input information





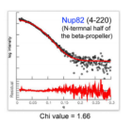
Consistent

3. Resampling

jackknifing
bootstrapping
cross-validation

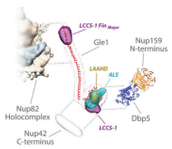
Low variance

4. Fit to data not used in modeling



Consistent

5. Biological sense

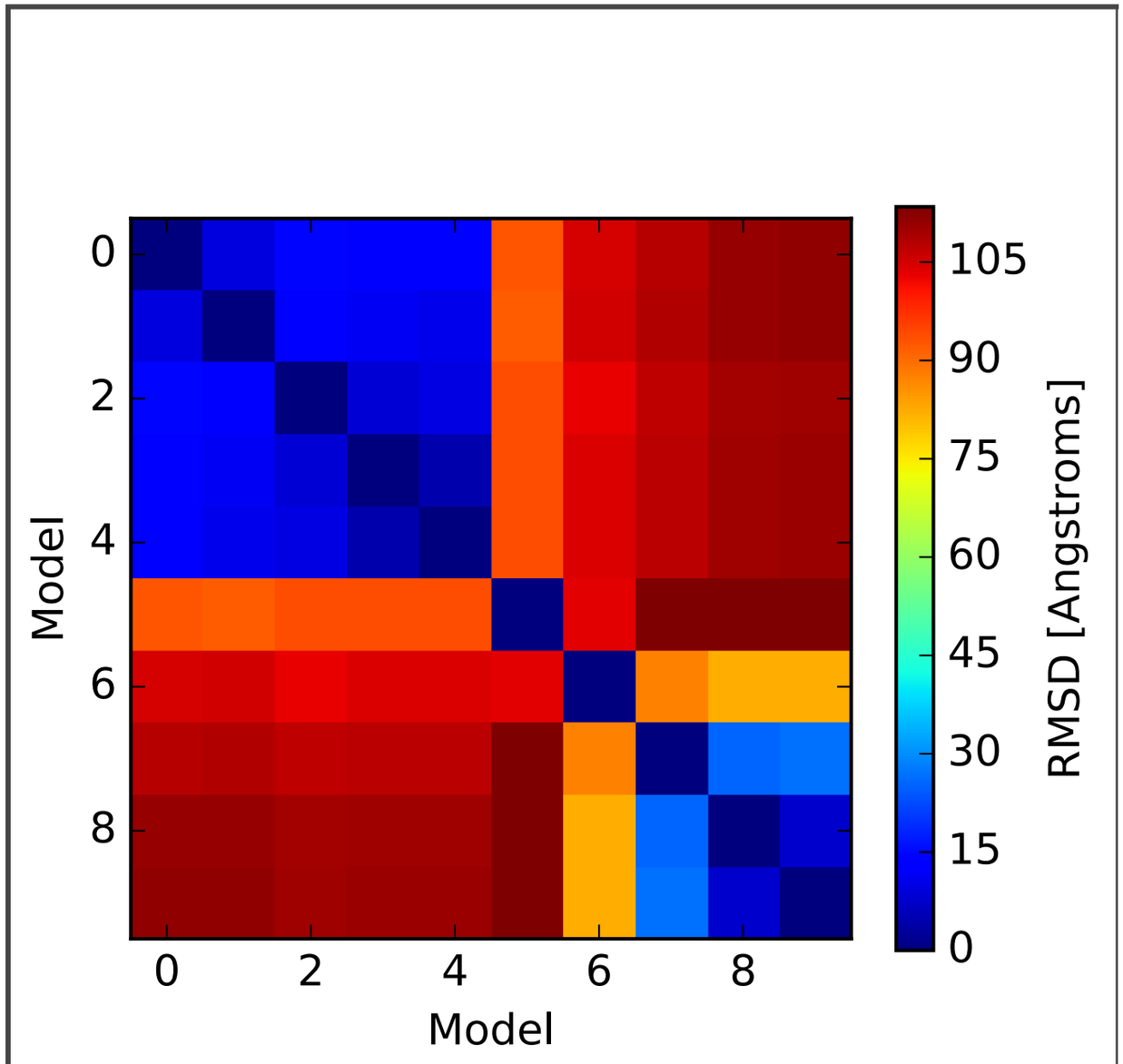


Yes

Validated Model

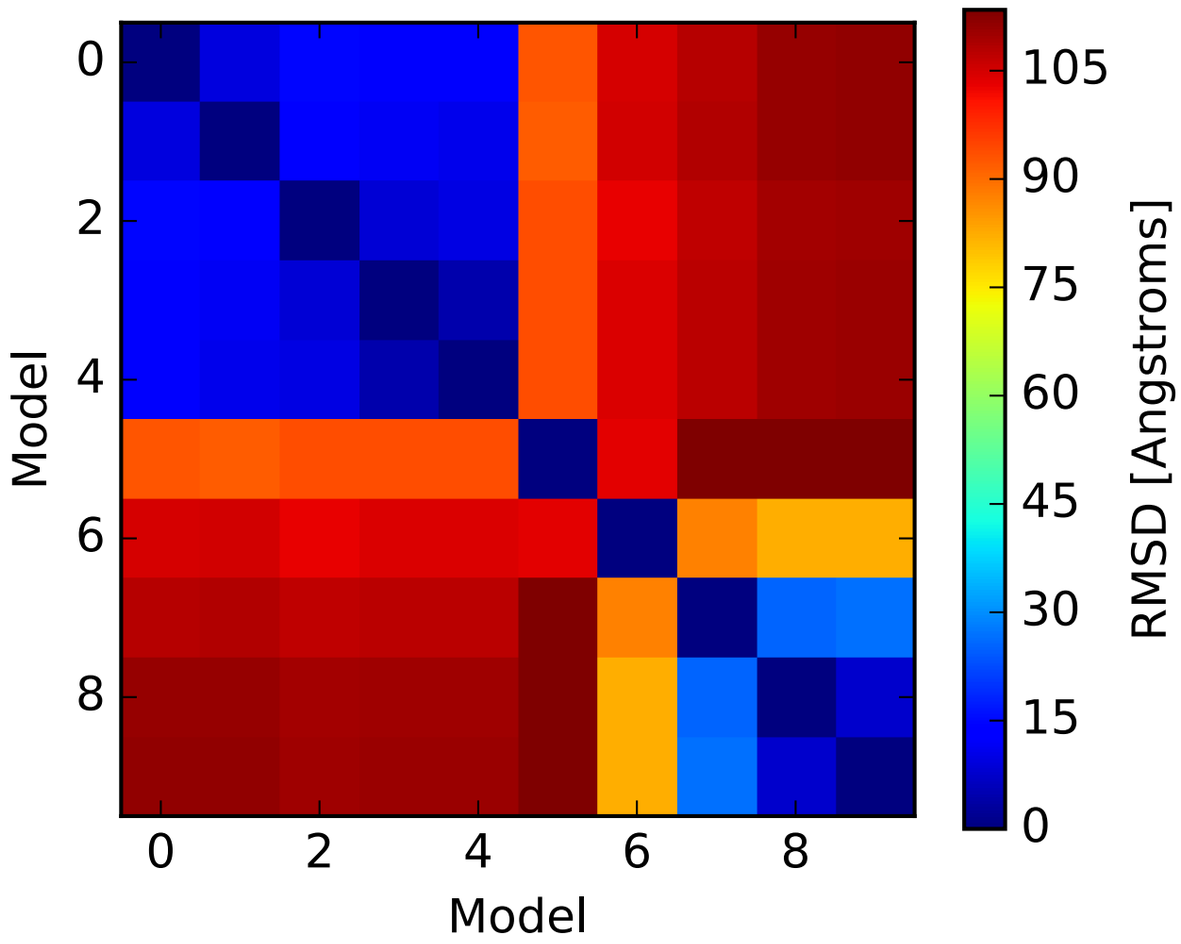
Clustering

Clustering

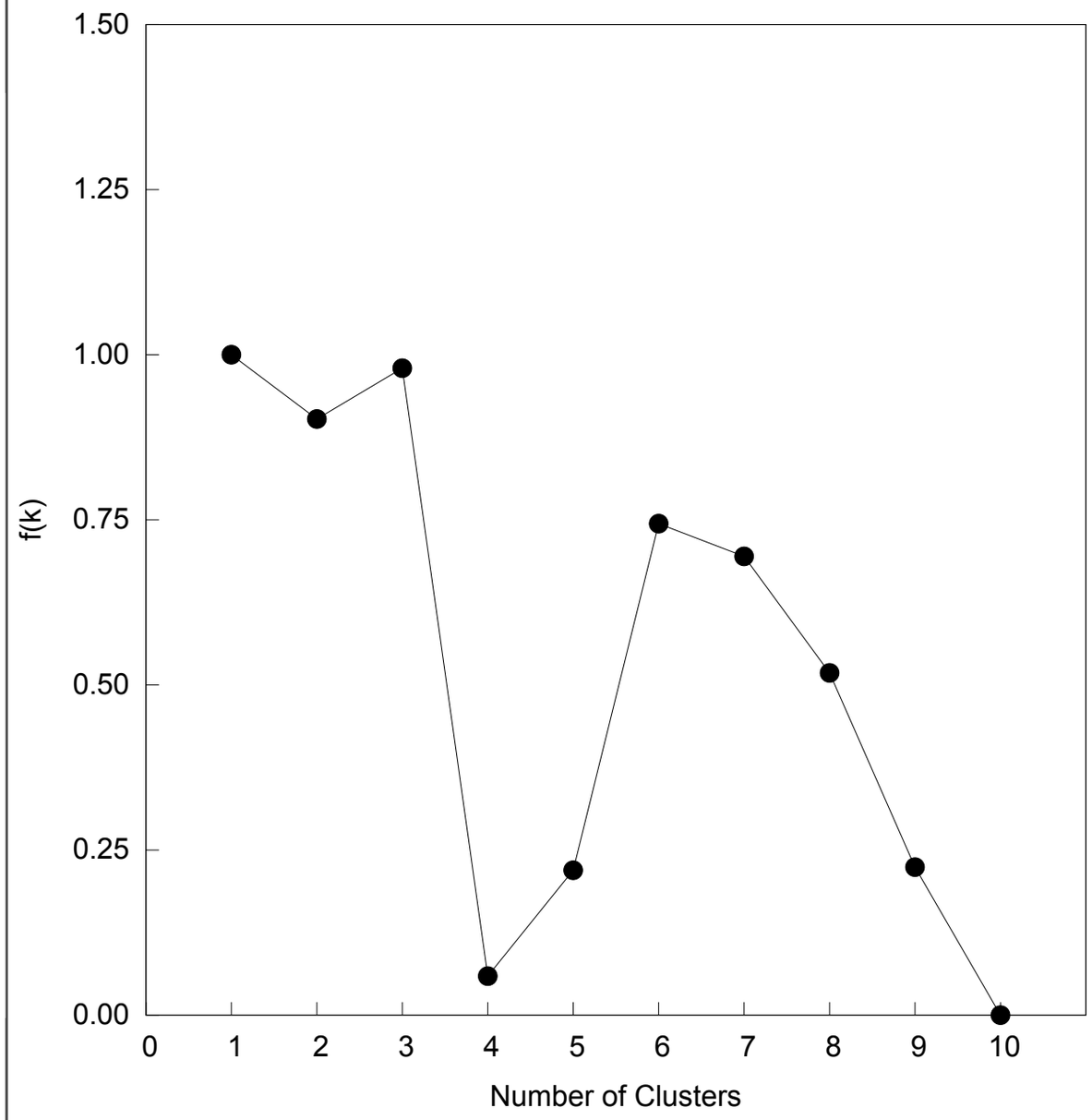


All vs. all RMSD for the top scoring models

Clustering



All vs. all RMSD for the top scoring models

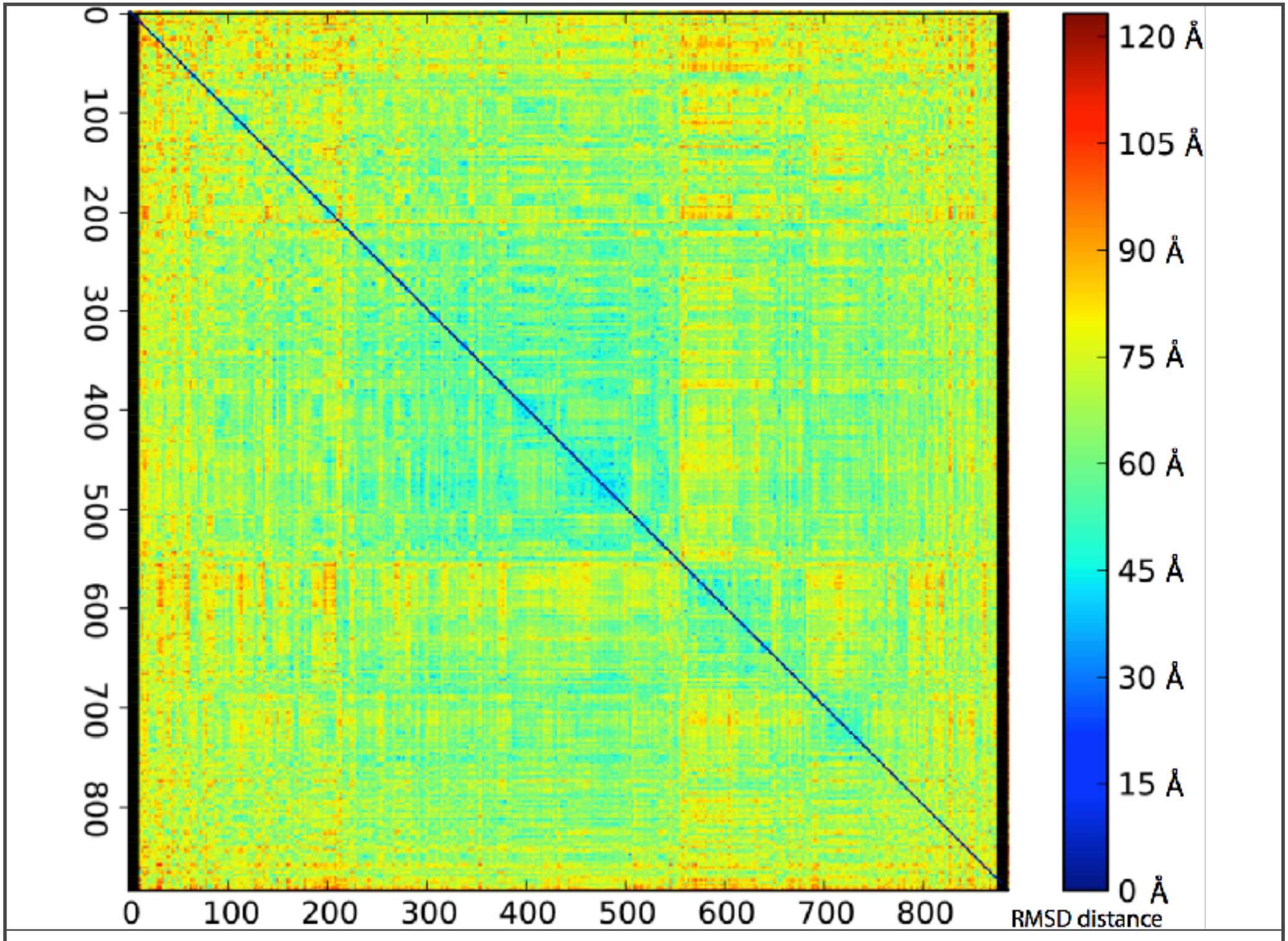


Determination of the optimal number of clusters using a distortion metric. Here, the extremum is four.

χ^2-contingency test for homogeneity of population

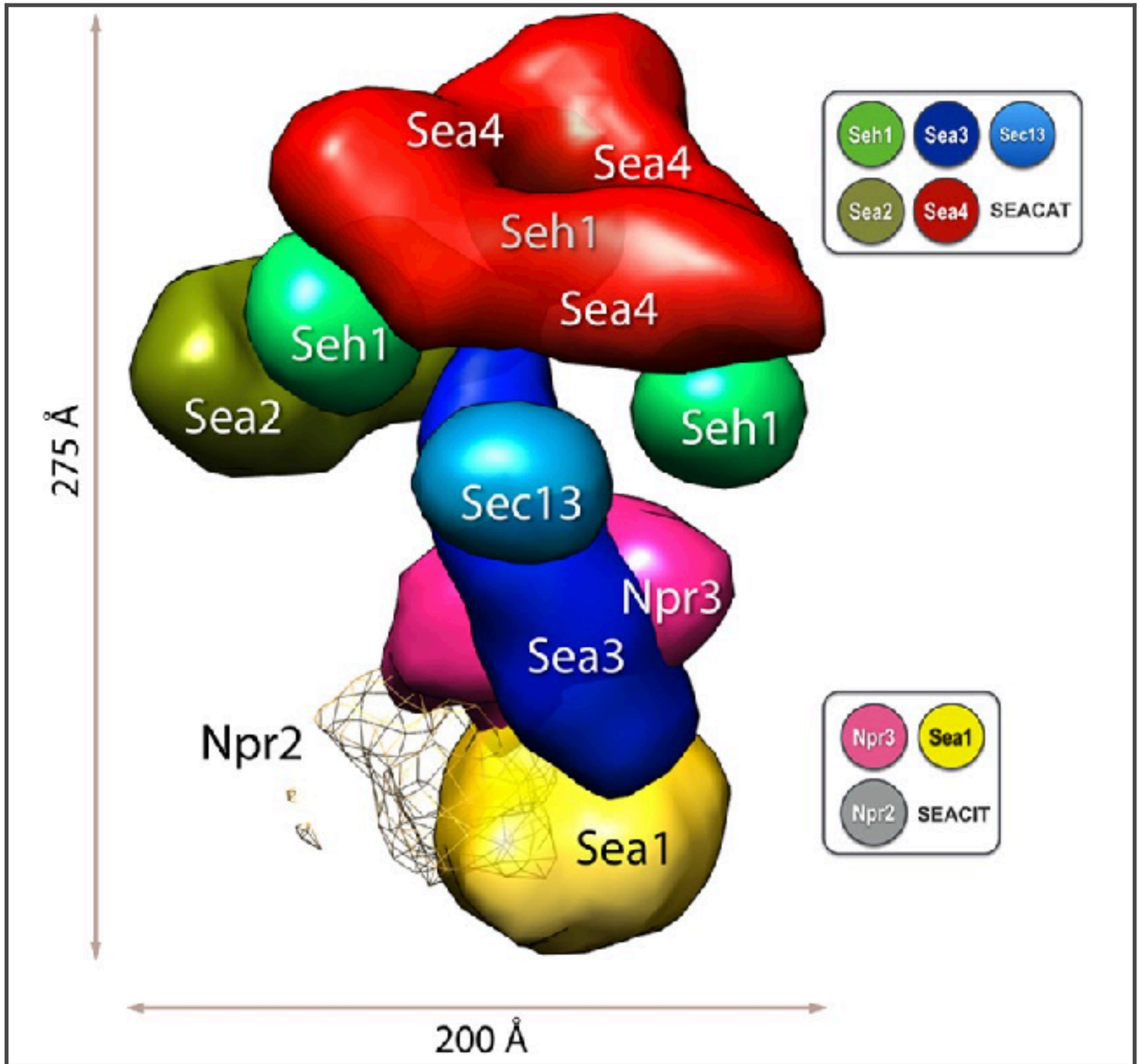
Sample 0	Sample 1	For such a small number of frames, obviously, the test fails. None of the cluster populations displays homogeneity, i.e. the two runs did not contribute equally to each clusters
0	100	
60	0	
20	0	Sampling has not converged
20	0	

Clustering for serious sampling



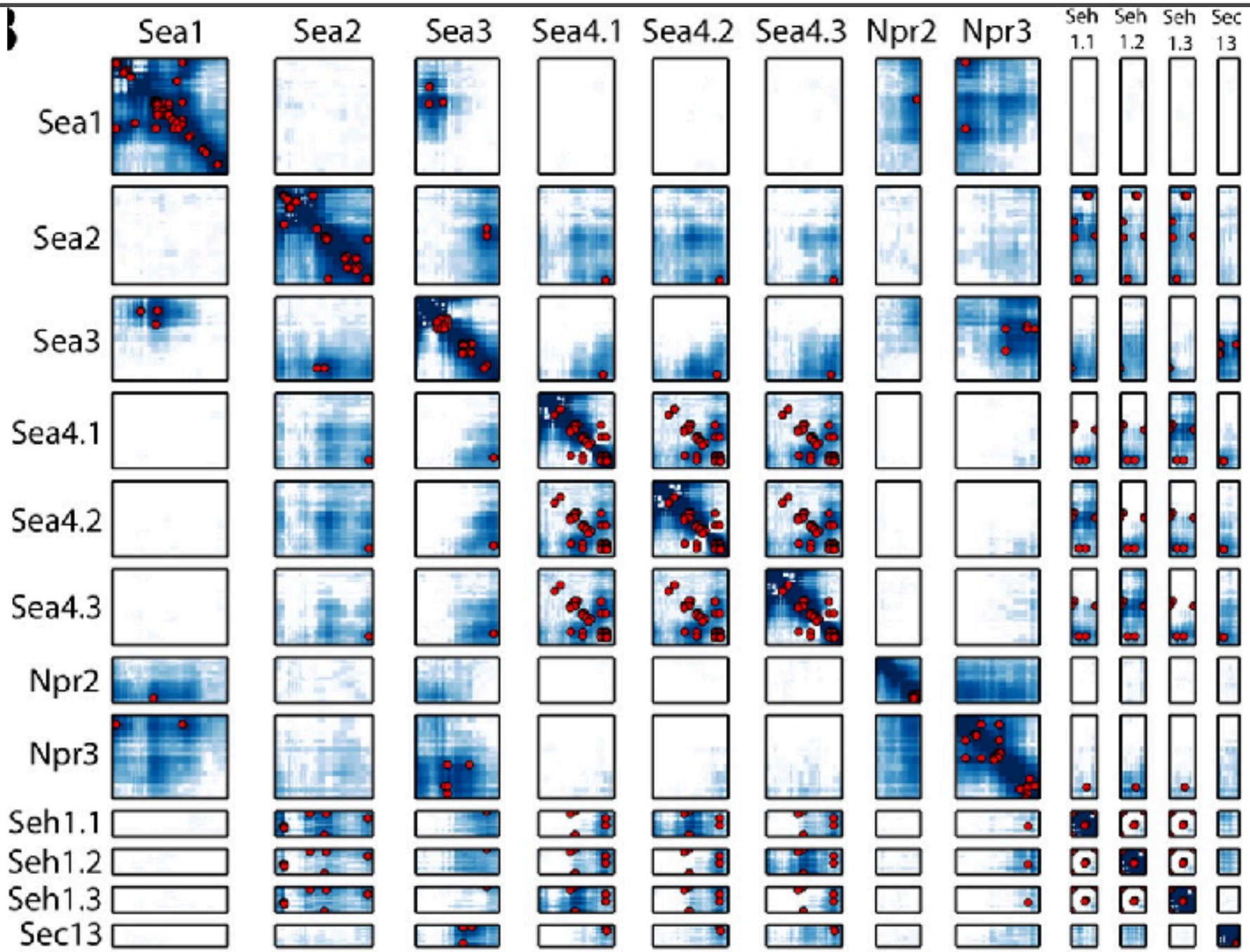
All vs. all RMSD for the top scoring models from 1000 independent trajectories, each rans for 50000 frames (total = $5 \cdot 10^7$ frames)

Probability Localization Density



Architecture of the SEA complex from the major cluster. The localization probability density of each component for the major cluster is displayed at 30% occupancy.

Satisfaction of input data



Contact map for the SEA complex resulting from the major cluster. The proximities of any two residues in the molecular architecture were measured by their relative contact frequency. A contact between a pair of residues was defined as an instance when their corresponding bead surfaces were less than 30 Å from each other. Cross-links are plotted as red dots, and the residue contact frequency is indicated by a color ranging from white (0) to dark blue (1). Each box contains the contact frequency between the corresponding pair of SEA complex proteins