

# Introduction to integrative modeling of protein assemblies with IMP

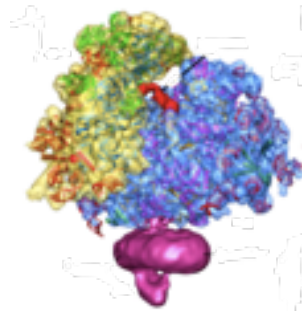
<http://salilab.org/>

Dr. Benjamin Webb

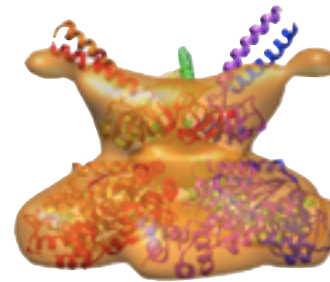
Sali Lab

University of California San Francisco

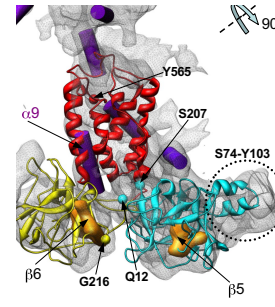
# Modeling structures



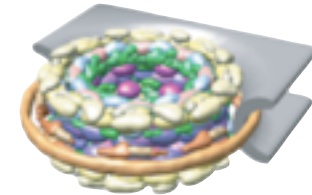
ribosome



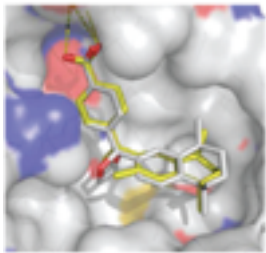
26S proteasome



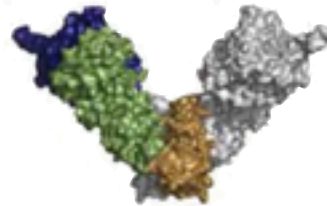
RyR1



NPC



RXRa



HSP90

atom positions

residue positions

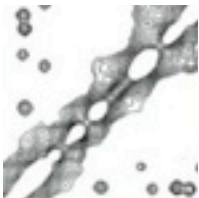
member  
orientations

member  
positions

Need to be able to simultaneously  
process structure on all these scales.



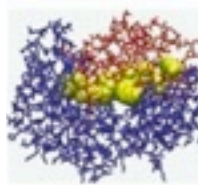
# Data sources



NMR



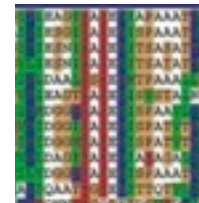
structure  
prediction



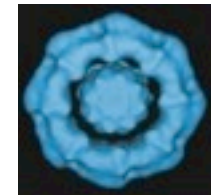
site-directed  
mutagenesis



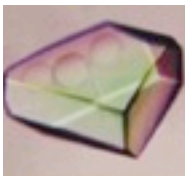
affinity  
purification



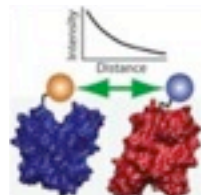
bioinformatics



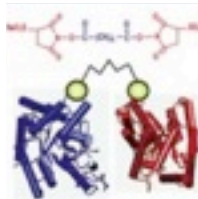
cryo-EM



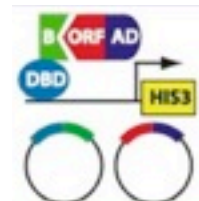
X-ray  
structures



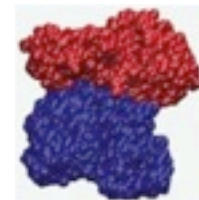
FRET



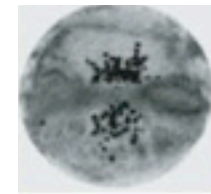
crosslinking



yeast  
two-hybrid



computational  
docking



immuno-EM

atom positions

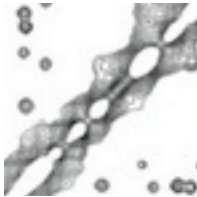
residue positions

member  
orientations

member  
positions

Need to be able to simultaneously  
process all the different data sources.

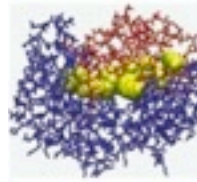
# Integrative modeling



NMR



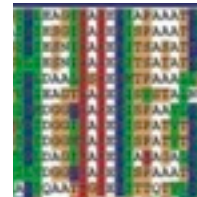
structure  
prediction



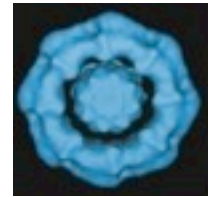
site-directed  
mutagenesis



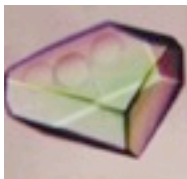
affinity  
purification



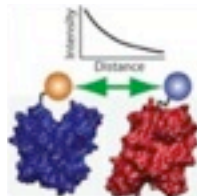
bioinformatics



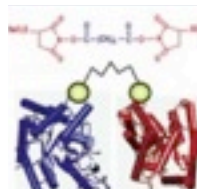
cryo-EM



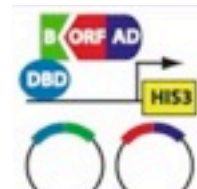
X-ray  
structures



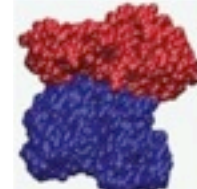
FRET



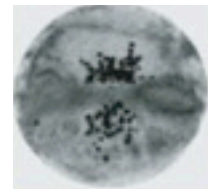
crosslinking



yeast  
two-hybrid



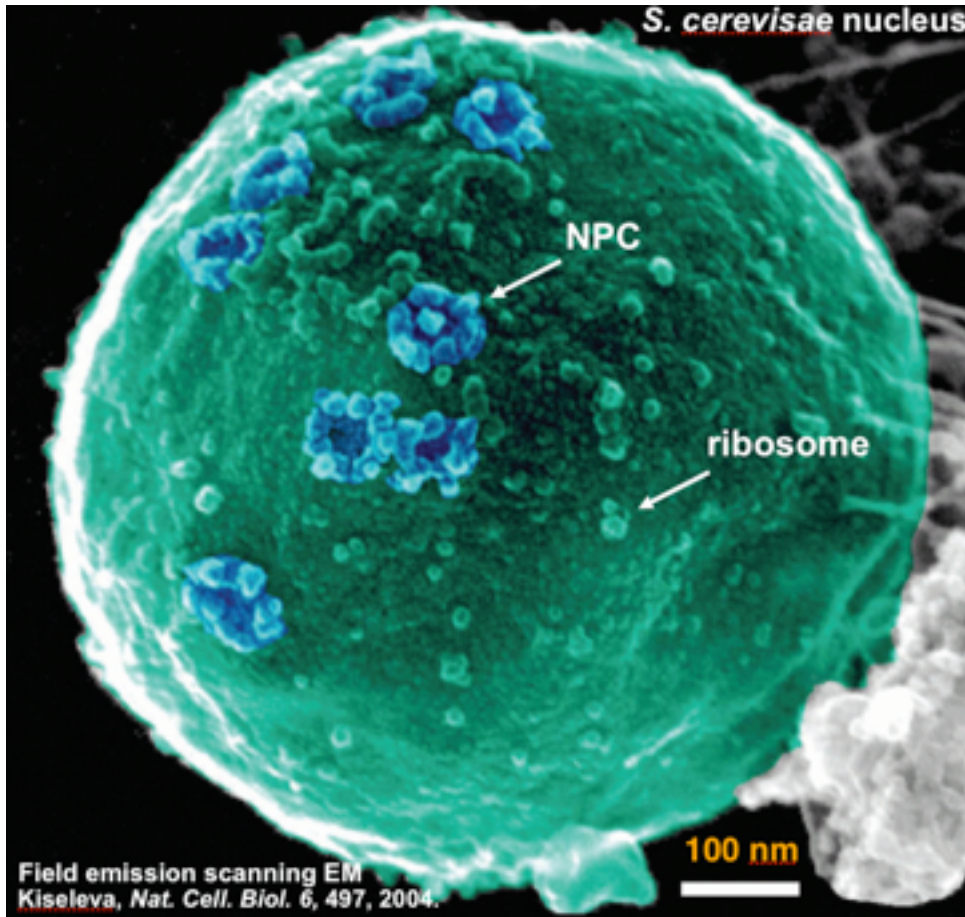
computational  
docking



immuno-EM

- Modeller uses only a subset of these data sources
- Limited to building structures of proteins at atomic resolution
- Our new Integrative Modeling Platform (IMP) package can use all data sources to build models of protein assemblies at a range of resolutions

# Nuclear Pore Complex (NPC)



Consists of broadly conserved **nucleoporins** (nups).

50 MDa complex: **~480** proteins of **30** different types.

**Mediates** all known nuclear **transport**, *via* cognate transport factors.

**Mike Rout**

Svetlana Dokudovskaya, Liesbeth Veenhoff  
Orit Karni-Schmidt, Julia Kipper, Tari Suprpto,  
Julia Kipper

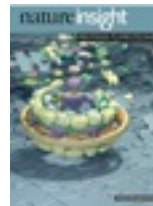
**Brian Chait**

Wenzhu Zhang, Rosemary Williams

**Rockefeller University**



Alber *et al.* Nature 450, 683-694, 2007  
Alber *et al.* Nature 450, 695-701, 2007  
Devos *et al.* PNAS 14, 2172-2177, 2006  
Devos *et al.* PLoS Biology 12, 1-9, 2004

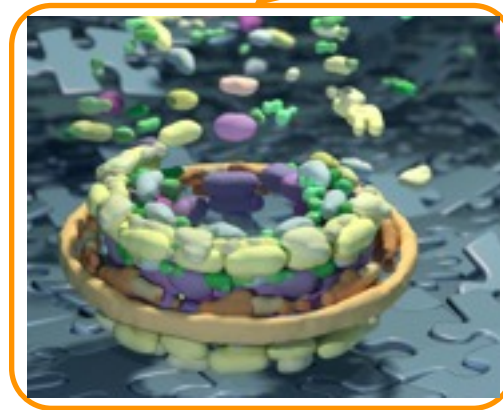


**Andrej Sali**

**Frank Alber**, Damien Devos  
Narayanan Eswar, Marc Marti-Renom

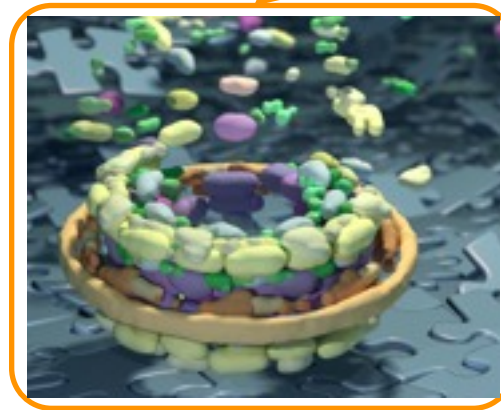
**UCSF**

# Configuration of proteins in NPC?



# Configuration of proteins in NPC?

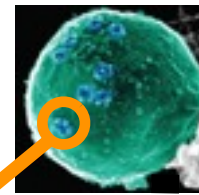
Use all information



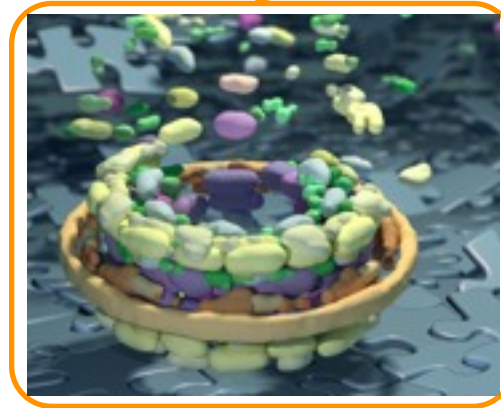
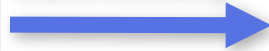


# Configuration of proteins in NPC?

Use all information

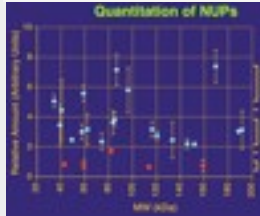


Protein  
Stoichiometry



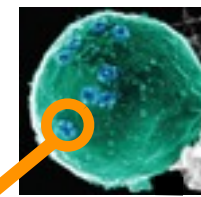
Quantitative  
Immunoblotting

30 relative  
abundances

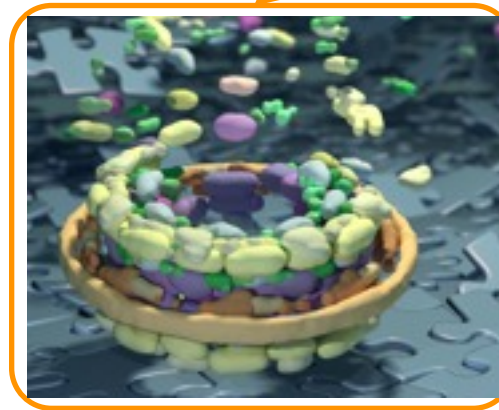


# Configuration of proteins in NPC?

Use all information



Protein  
Stoichiometry

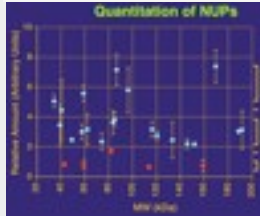


Protein  
Shape



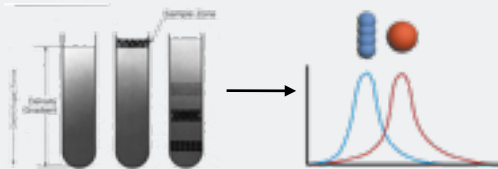
Quantitative  
Immunoblotting

30 relative  
abundances



Ultracentrifugation

30 S-values 1 S-value



Bioinformatics and  
Membrane  
Fractionation

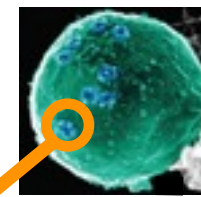
30 protein  
sequences

LEAGIHAEETAP  
ESGHAETISP  
EENHAEETIS  
EENHAEETIS  
LDAAHSEETP  
LEAGHAEETIS  
MDGHAETISP

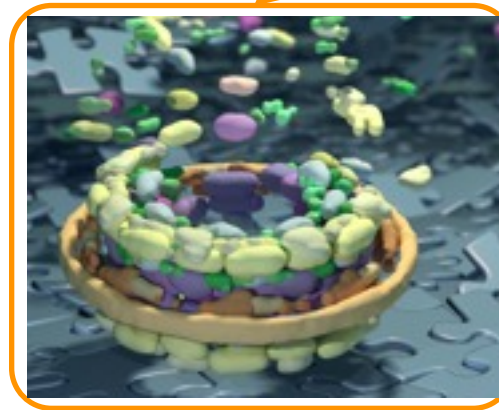


# Configuration of proteins in NPC?

Use all information



Protein  
Stoichiometry

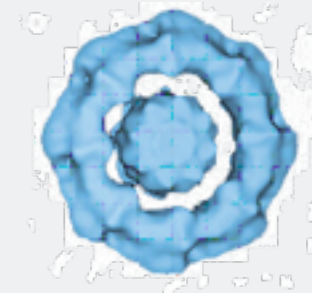


Symmetry



Electron  
Microscopy

electron microscopy  
map

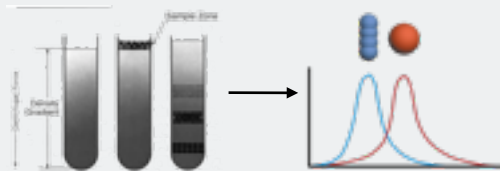


Protein  
Shape



Ultracentrifugation

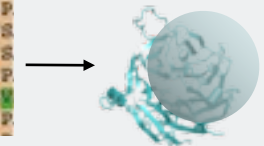
30 S-values 1 S-value



Bioinformatics and  
Membrane  
Fractionation

30 protein  
sequences

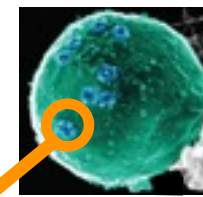
LEAGIHAEVETAP  
ESGIIHAEVETSP  
EENIHAEVETTS  
EENIHAEVETTS  
LDAAHSEETMT  
LEAGIHAEVETSP  
MDGCHAEVETSP





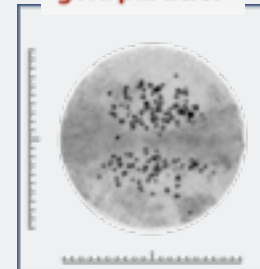
# Configuration of proteins in NPC?

## Use all information



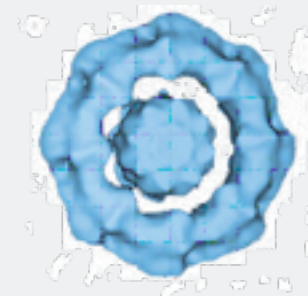
Protein  
Localization

Immuno-  
Electron  
Microscopy  
10,615  
gold particles



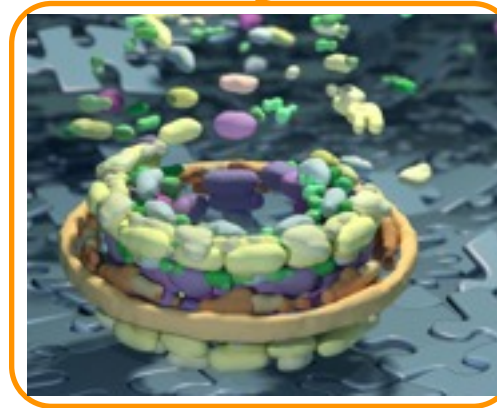
Electron  
Microscopy

electron microscopy  
map



Symmetry

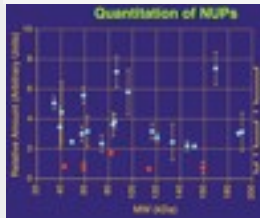
Protein  
Shape



Protein  
Stoichiometry

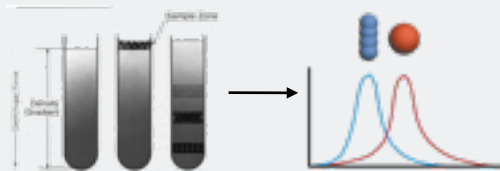
Quantitative  
Immunoblotting

30 relative  
abundances



Ultracentrifugation

30 S-values 1 S-value



Bioinformatics and  
Membrane  
Fractionation

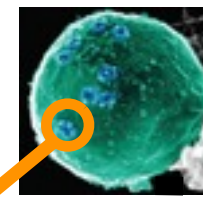
30 protein  
sequences

LEAGIHAEETAP  
ESGIAHEETSP  
EENIHAEETTS  
EENIHAEETTS  
LDAALSHETMT  
LEAGIHAEETSP  
MDGGAHEETSP



# Configuration of proteins in NPC?

## Use all information



Protein  
Stoichiometry

Protein  
Localization

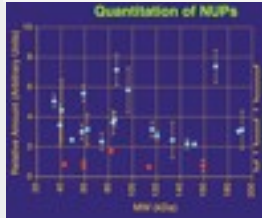
Protein-protein  
Proximities

Symmetry

Protein  
Shape

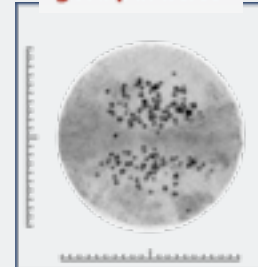
Quantitative  
Immunoblotting

30 relative  
abundances



Immuno-  
Electron  
Microscopy

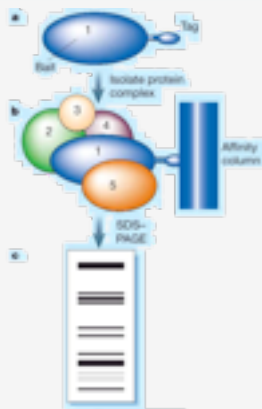
10,615  
gold particles



Affinity  
Purification

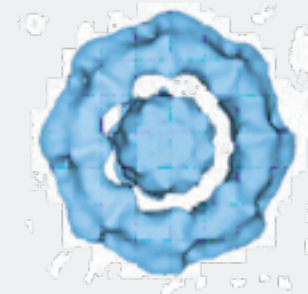
Overlay  
Assay

75 composites 7 contacts



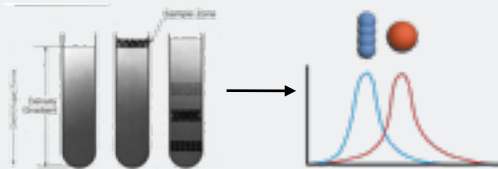
Electron  
Microscopy

electron microscopy  
map



Ultracentrifugation

30 S-values 1 S-value



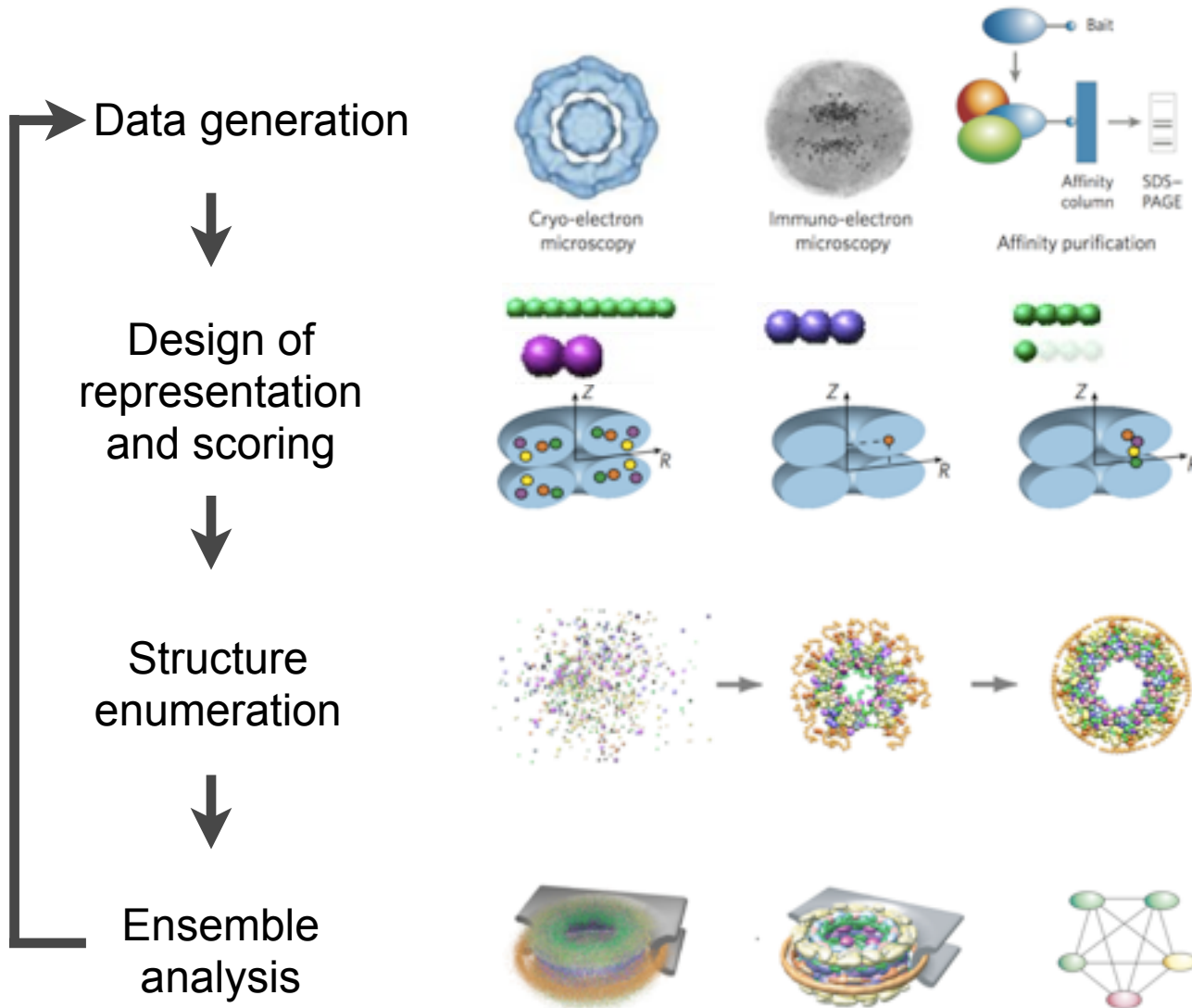
Bioinformatics and  
Membrane  
Fractionation

30 protein  
sequences

LEAGIHAEVETAP  
ESGIIHAEVETAP  
ESNIIHAEVETAP  
ESNIIHAEVETAP  
ESNIIHAEVETAP  
ESNIIHAEVETAP  
ESNIIHAEVETAP  
ESNIIHAEVETAP  
ESNIIHAEVETAP  
ESNIIHAEVETAP



# Integrative modeling - IMP



Alber *et al.* *Nature* 2007 • Robinson, Sali, Baumeister. *Nature* 2007 •  
Russel, et al. *Current Opinion in Cell Biology*, 2009

# Representation



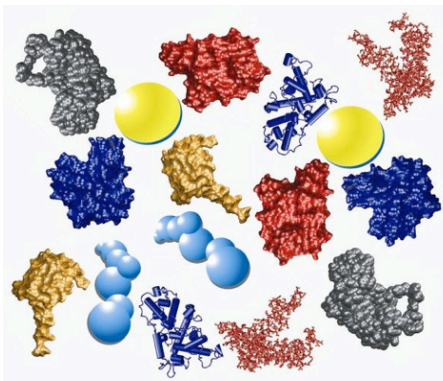
Atomic (crystal or homology model)



Protein as sphere



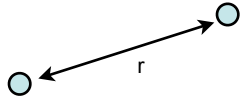
Domains as spheres (beads on a string)



Mixture

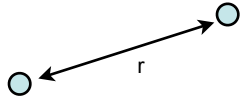
# Scoring (restraints)

# Scoring (restraints)



$$S = \frac{1}{2}k(r-m)^2, \quad dS/dx, \quad dS/dy, \quad dS/dz$$

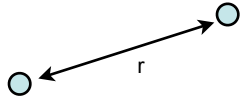
# Scoring (restraints)



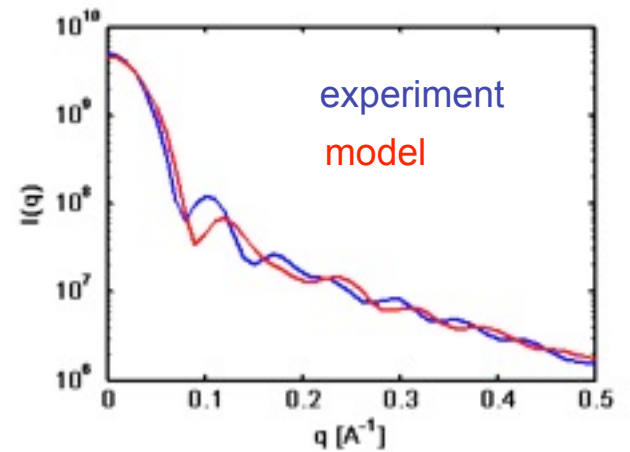
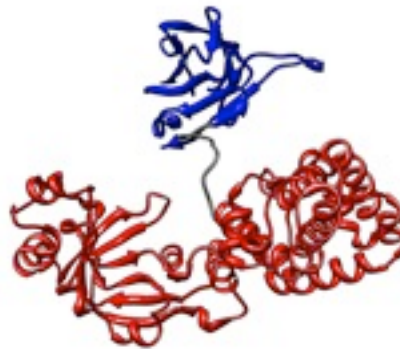
$$S = \frac{1}{2}k(r-m)^2, \quad dS/dx, \quad dS/dy, \quad dS/dz$$



# Scoring (restraints)



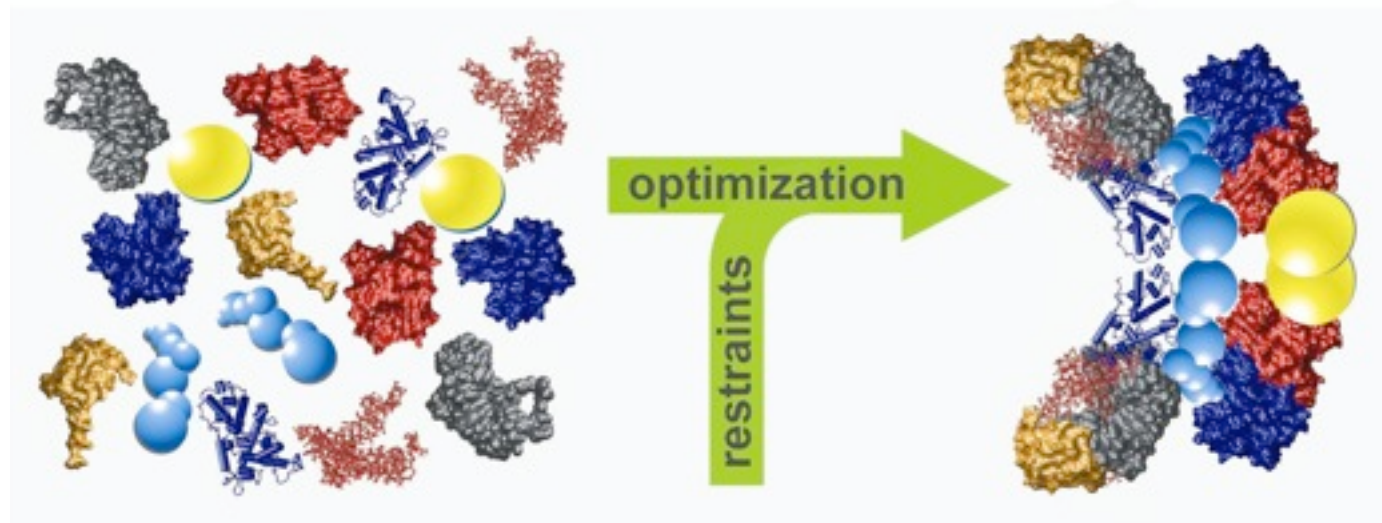
$$S = \frac{1}{2}k(r-m)^2, \quad dS/dx, \quad dS/dy, \quad dS/dz$$





# Optimization

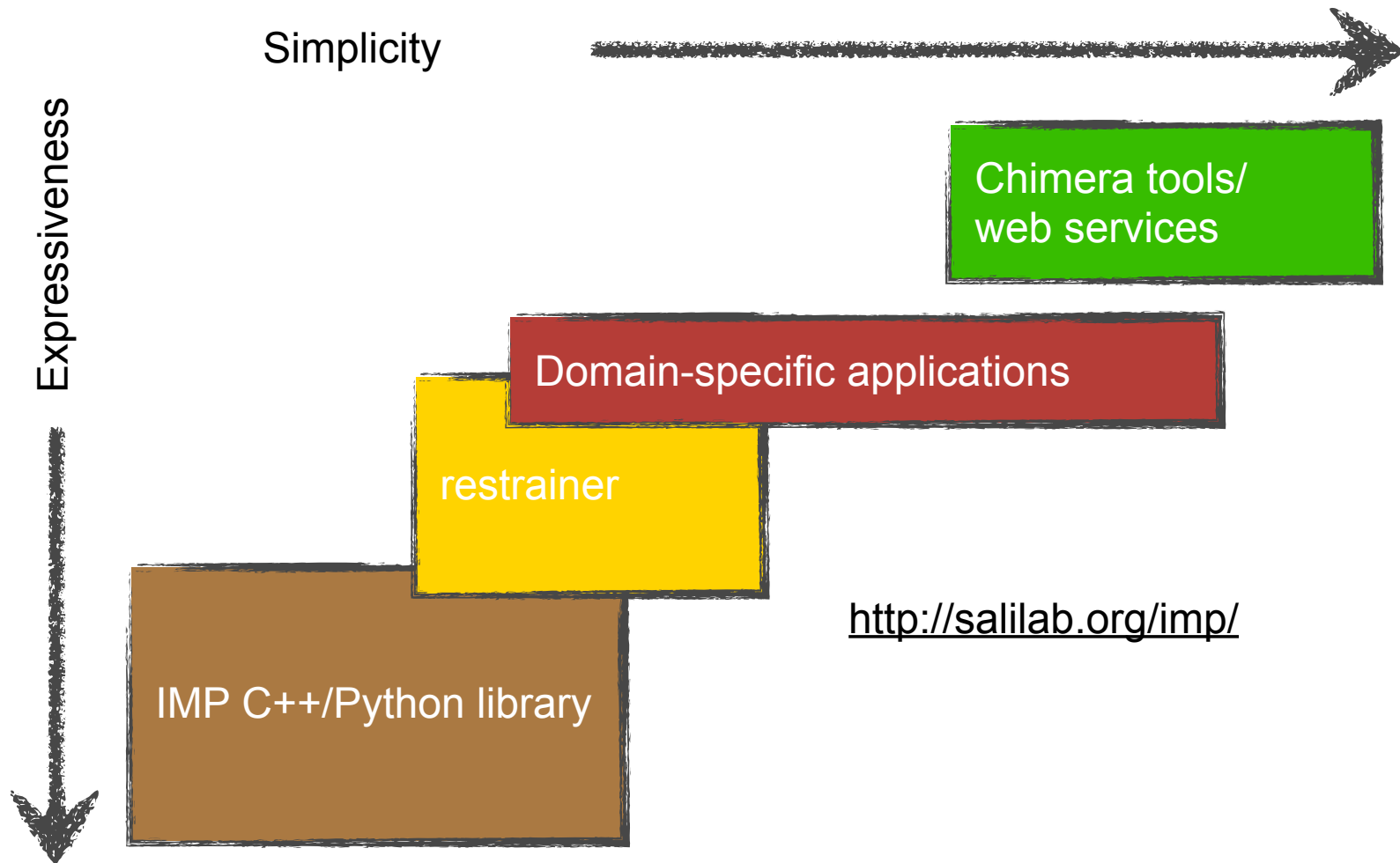
- Total score is sum of all restraints
- Score then minimized by moving the particles with
  - Conjugate gradients
  - Molecular dynamics
  - Monte Carlo
  - Inference



# Ensemble analysis

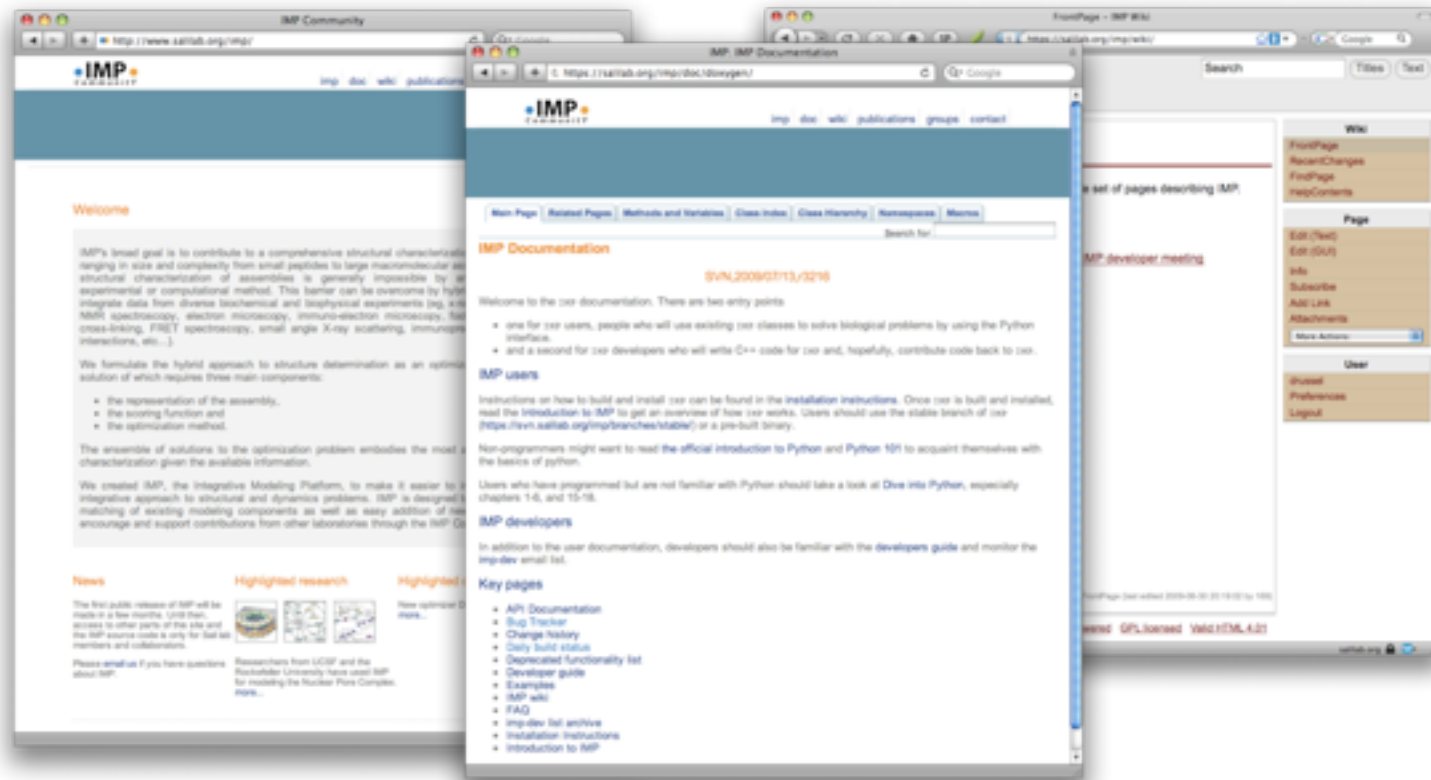
- How accurate is a model? *Alber et al. Nature 450, 695-702, 2007.*
  - Self-consistency of independent experimental data.
  - Structural similarity among the configurations in the ensemble that satisfy the input restraints.
  - Simulations where a native structure is assumed, corresponding restraints simulated from it, and the resulting calculated structure compared with the assumed native structure.
  - Patterns emerging from a mapping of independent and unused data on the structure that are unlikely to occur by chance.
  - Experimental spatial data that were not used in the calculation of the structure.

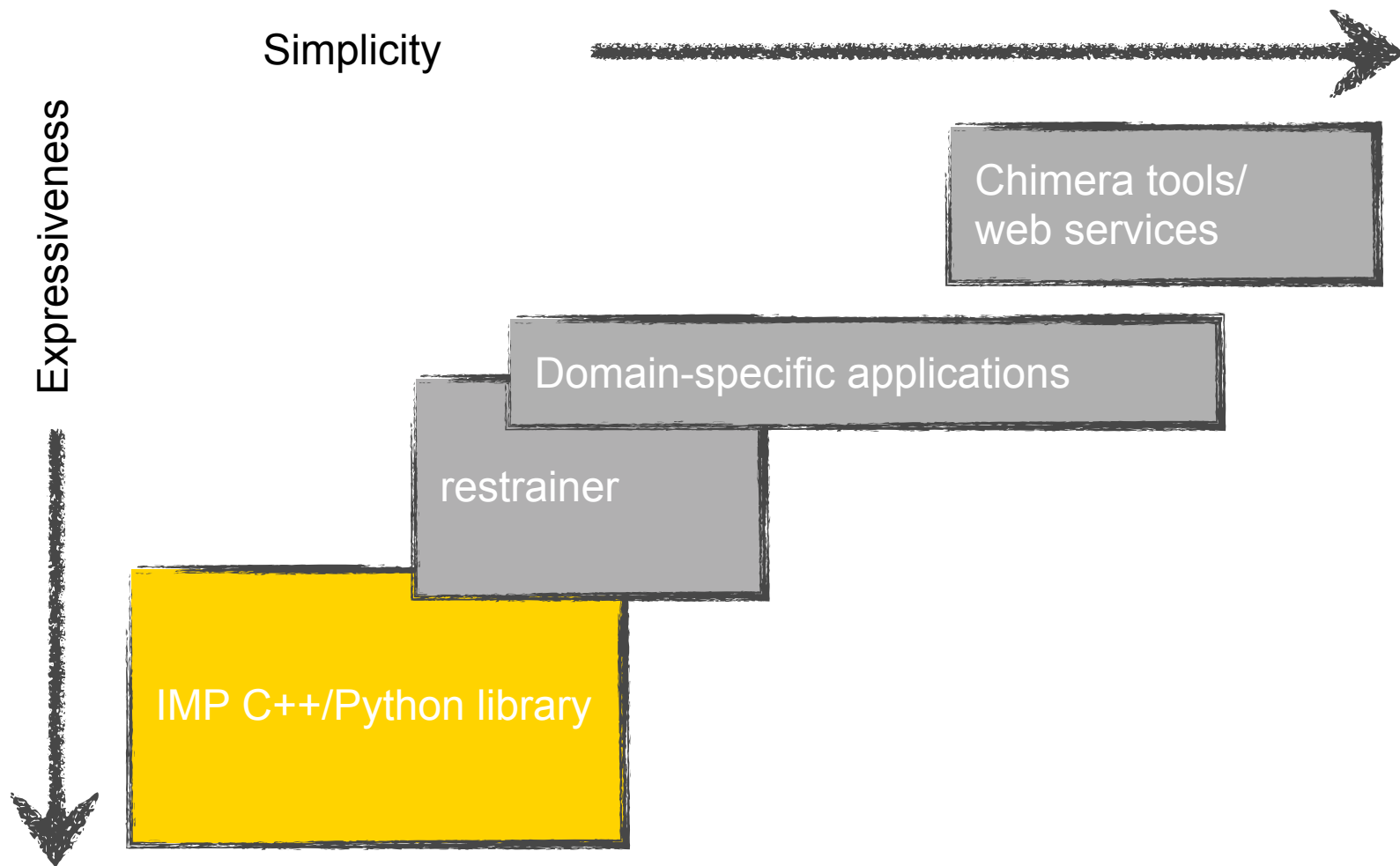
# Presenting IMP



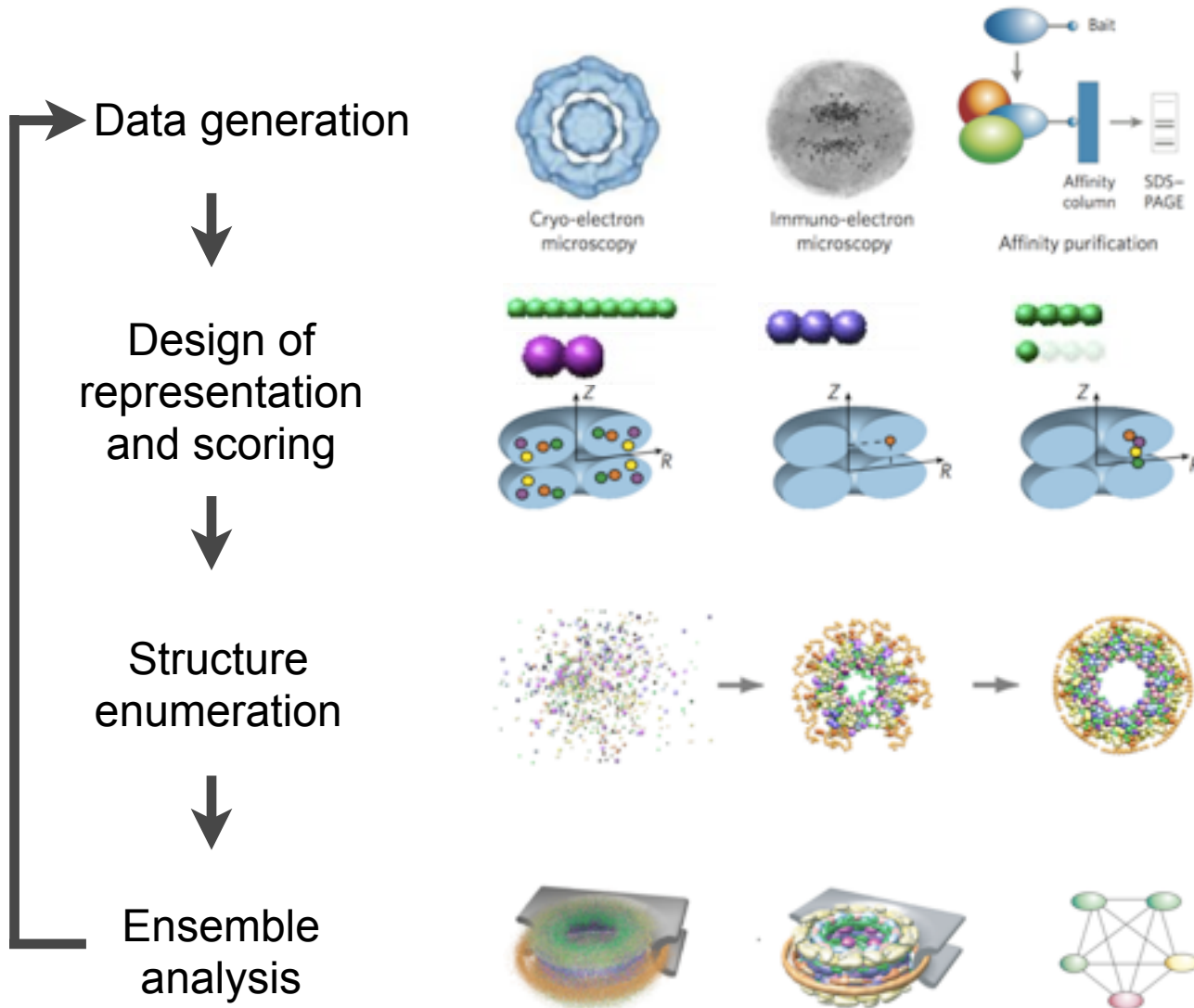
# IMP release

- IMP 1.0 is available as open source (LGPL)
- Binaries (Mac/Windows/Linux), source, SVN, documentation, wiki, examples, mailing lists, unit testing, bug tracking...



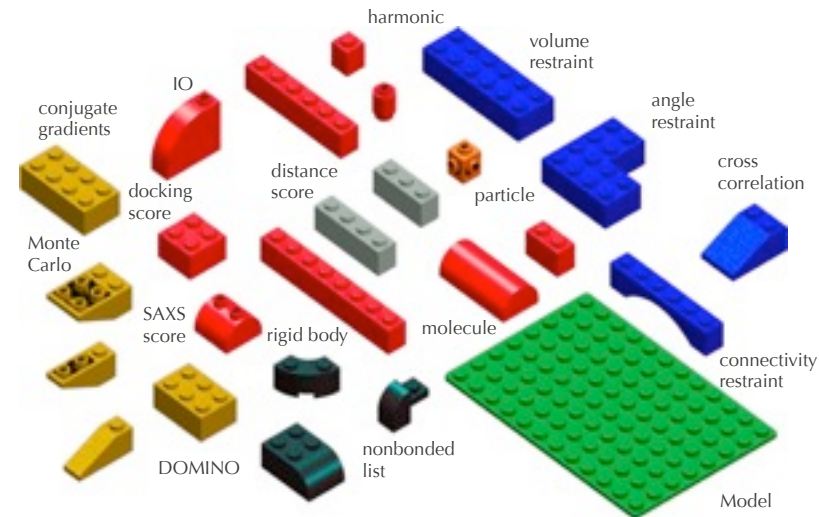
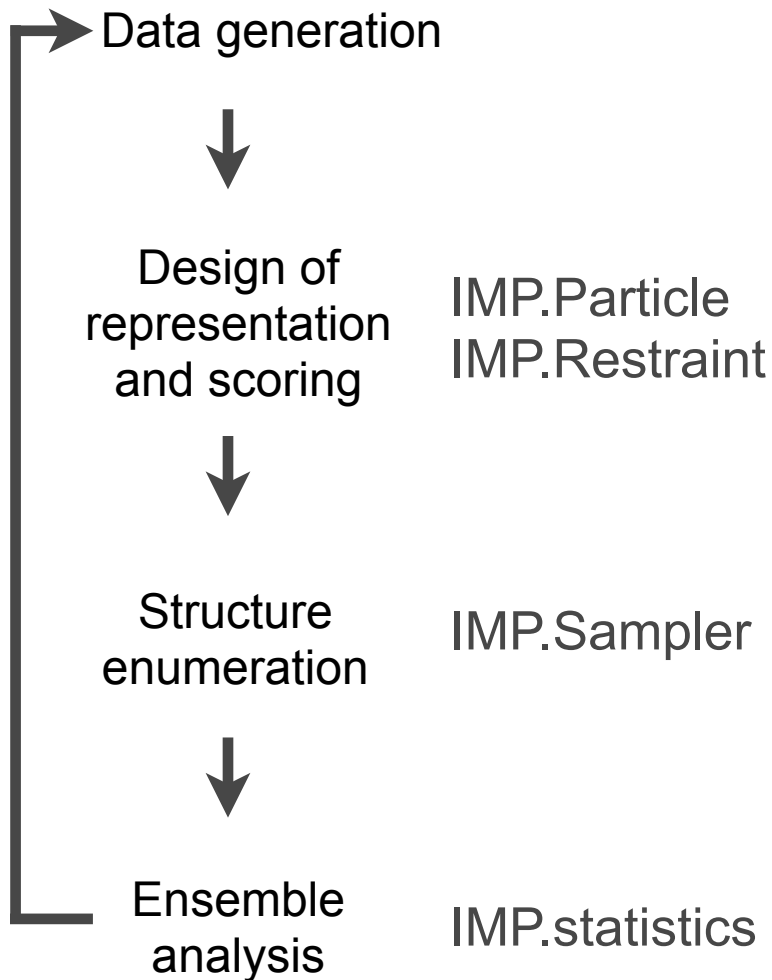


# C++/Python library

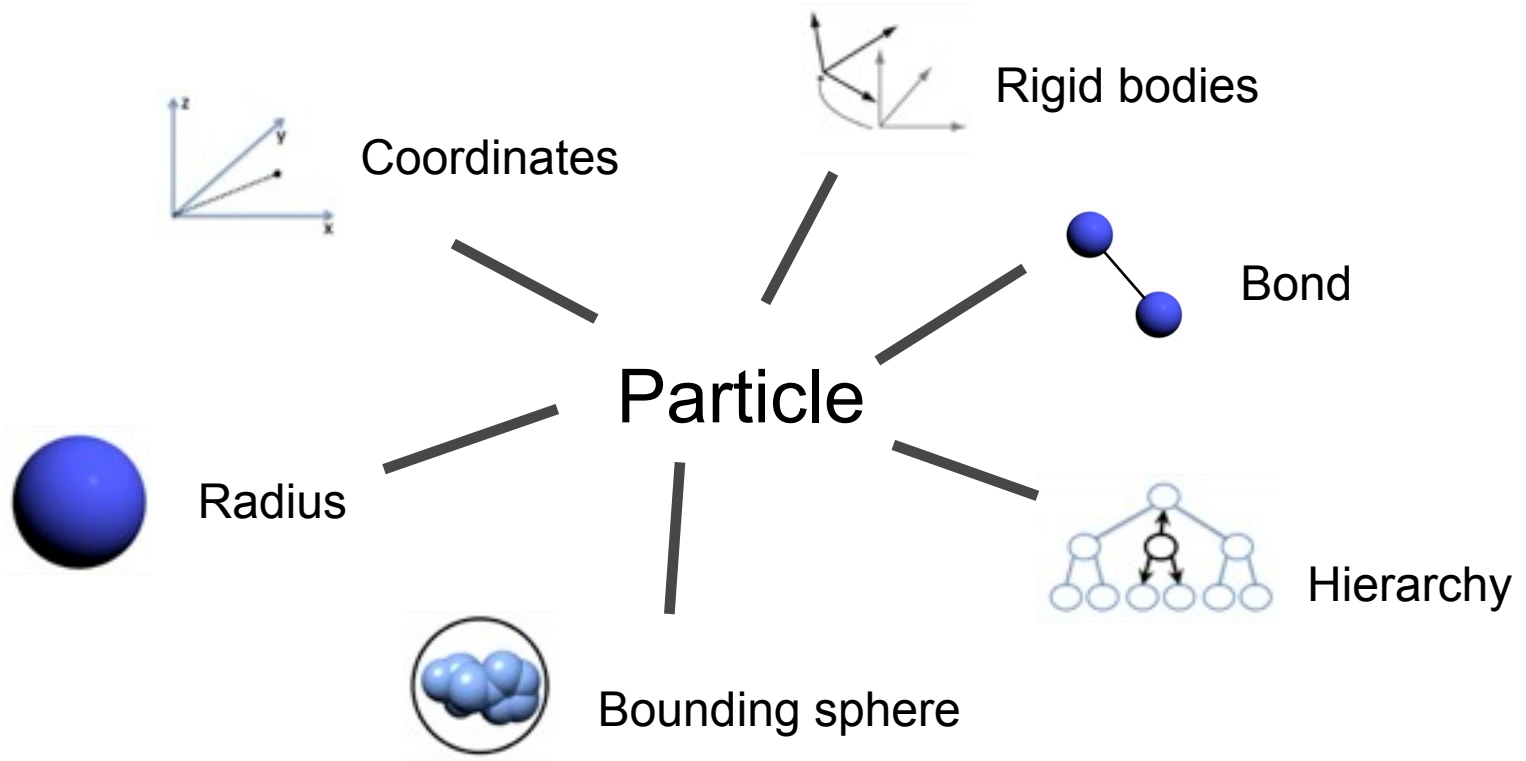


Alber *et al.* *Nature* 2007 • Robinson, Sali, Baumeister. *Nature* 2007 •  
Russel, et al. *Current Opinion in Cell Biology*, 2009

# C++/Python library



# Flexible classes

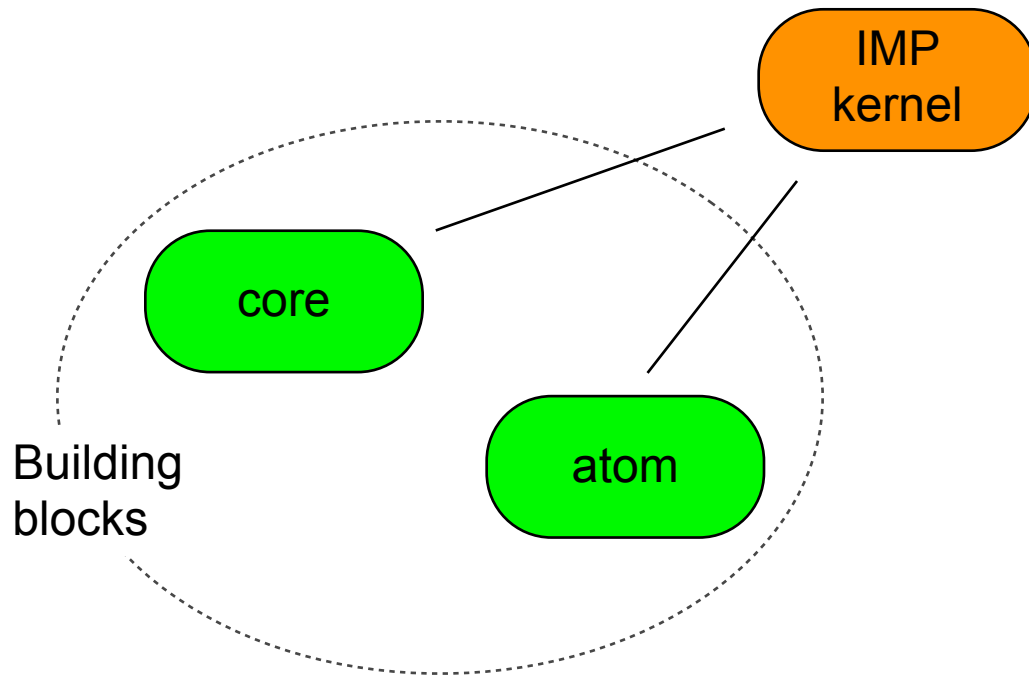




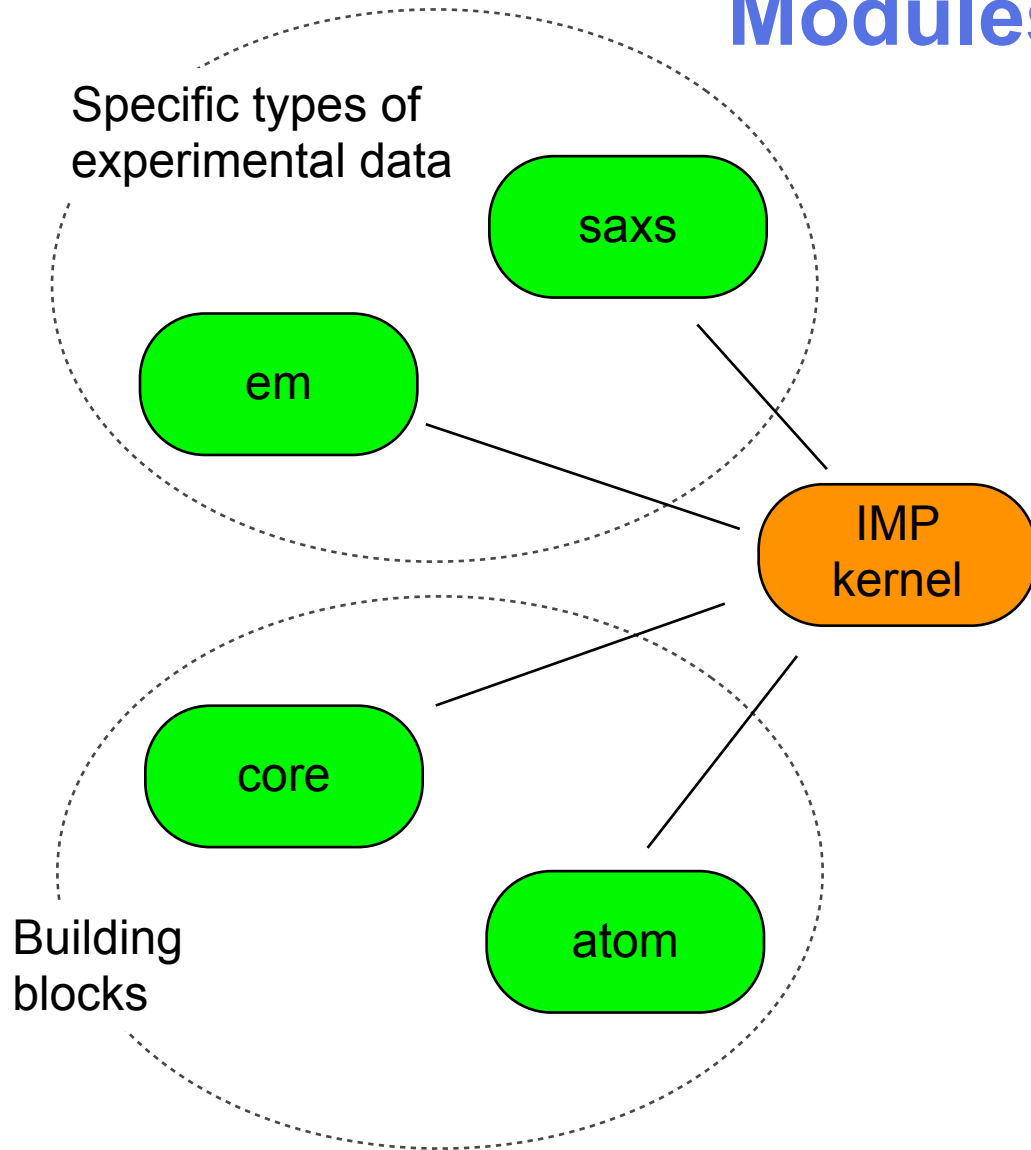
# Modules



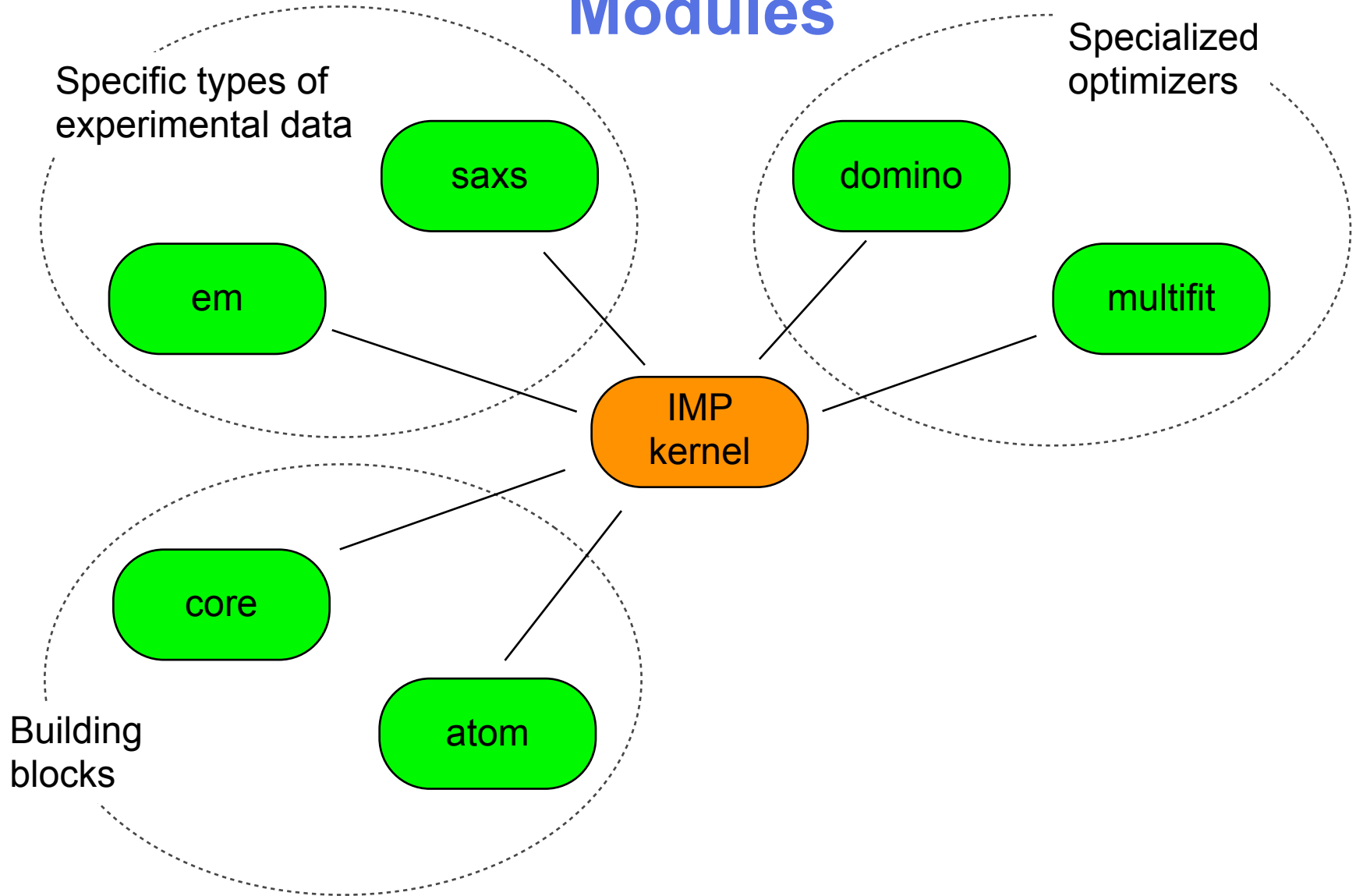
# Modules



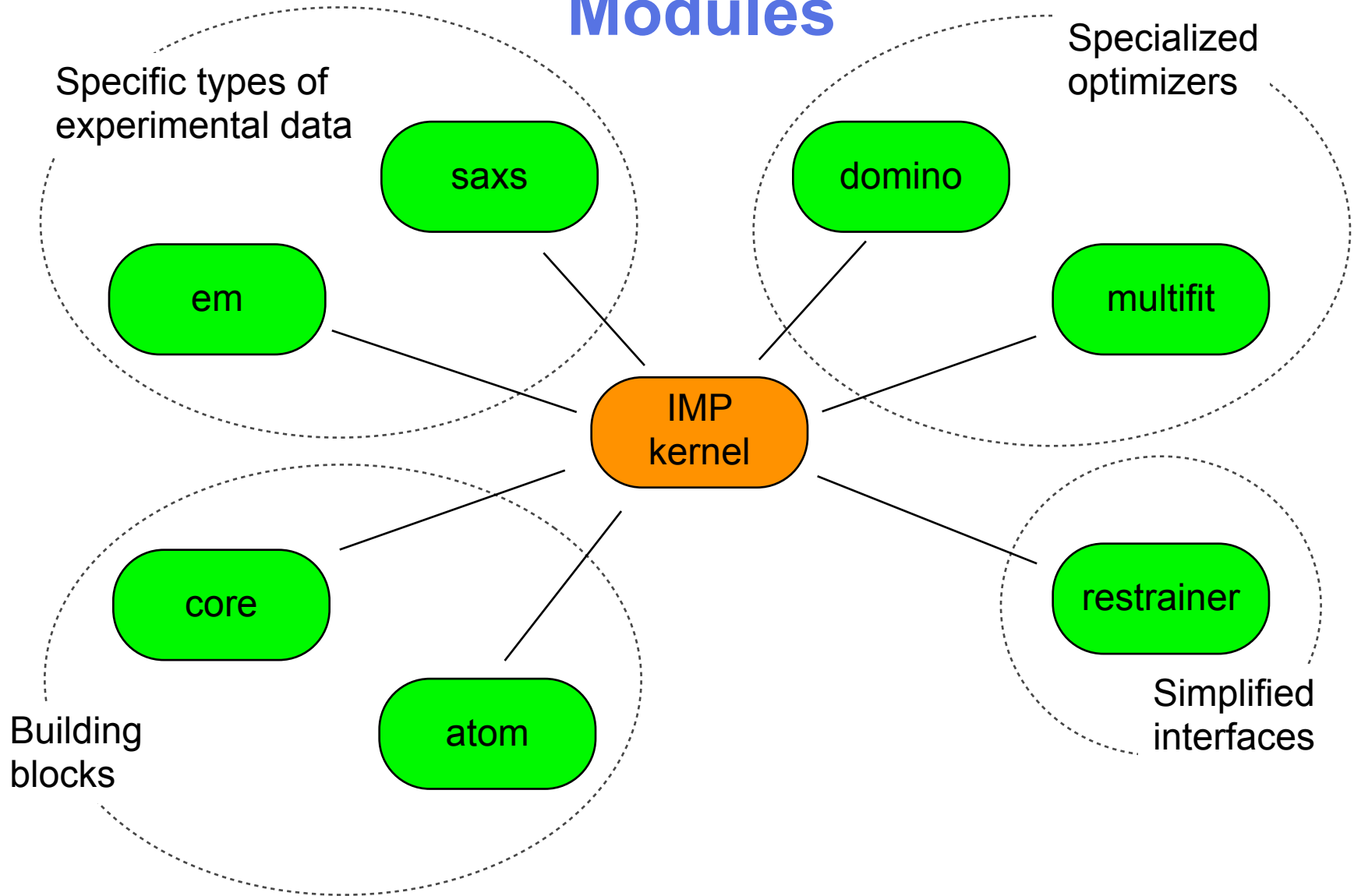
# Modules



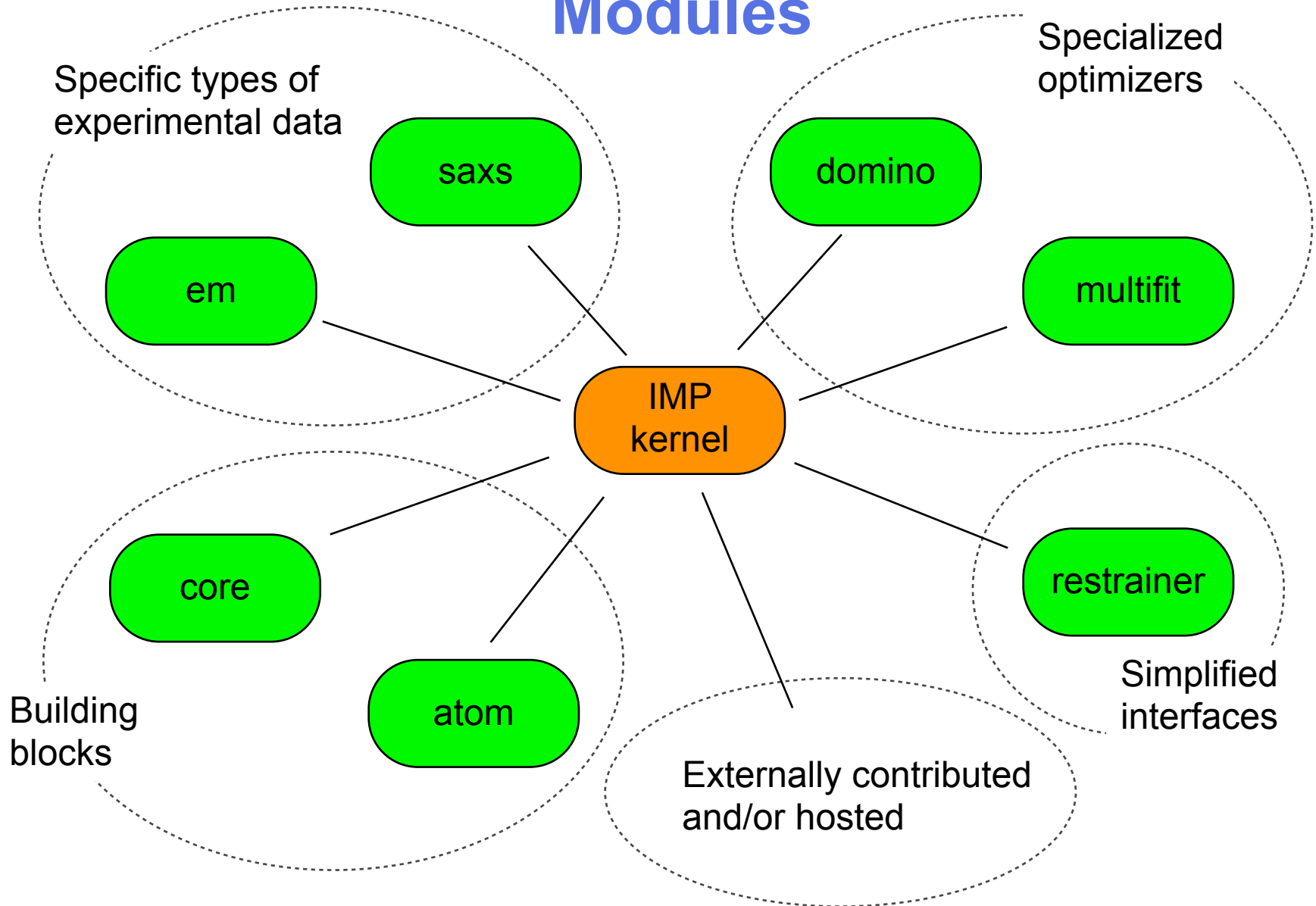
# Modules



# Modules



# Modules



# IMP kernel

- Does little by itself
- Provides abstract C++ classes to define interfaces between various parts of IMP
  - e.g. the kernel defines a “Restraint” as something which, given a set of Particles, returns a score on them - but doesn’t define any **actual** restraints
  - the ‘core’ module provides useful restraints, e.g. to restrain a distance between two point-like particles
- Every class can also be used from Python
- We can develop our own integrative modeling protocols from scratch by writing Python scripts
- Alternatively, applications may link against the C++ library or import the Python modules

# Scripting example

- `simple.py` is a simple IMP Python script
- First part; IMP and system setup:

```
import IMP
import IMP.algebra
import IMP.core

m = IMP.Model()

# Create two "untyped" Particles
p1 = IMP.Particle(m)
p2 = IMP.Particle(m)

# "Decorate" the Particles with x,y,z attributes (point-like particles)
d1 = IMP.core.XYZ.setup_particle(p1)
d2 = IMP.core.XYZ.setup_particle(p2)

# Use some XYZ-specific functionality (set coordinates)
d1.set_coordinates(IMP.algebra.Vector3D(10.0, 10.0, 10.0))
d2.set_coordinates(IMP.algebra.Vector3D(-10.0, -10.0, -10.0))
print d1, d2
```



# Low-level scripting example

- Second part; add restraints:

```
# Harmonically restrain p1 to be zero distance from the origin
f = IMP.core.Harmonic(0.0, 1.0)
s = IMP.core.DistanceToSingletonScore(f, IMP.algebra.Vector3D(0., 0., 0.))
r1 = IMP.core.SingletonRestraint(s, p1)
m.add_restraint(r1)

# Harmonically restrain p1 and p2 to be distance 5.0 apart
f = IMP.core.Harmonic(5.0, 1.0)
s = IMP.core.DistancePairScore(f)
r2 = IMP.core.PairRestraint(s, IMP.ParticlePair(p1, p2))
m.add_restraint(r2)
```

# Low-level scripting example

- Final part; generate a system configuration consistent with all restraints:

```
# Optimize the x,y,z coordinates of both particles with conjugate gradients
d1.set_coordinates_are_optimized(True)
d2.set_coordinates_are_optimized(True)
o = IMP.core.ConjugateGradients(m)
o.optimize(50)
print d1, d2
```

- Run the final script like any other Python script

# C++ versus Python

- The IMP libraries are deliberately set up so as to be very similar to use between C++ and Python
- Main differences are
  - language syntax (e.g. 'import IMP' in Python roughly translates to '#include <IMP.h>' in C++)
  - memory handling (Python has automatic refcounting; in C++ this must be done explicitly using the `IMP::Pointer` class)

```
import IMP
import IMP.algebra
import IMP.core

m = IMP.Model()

# Create two "untyped" Particles
p1 = IMP.Particle(m)
p2 = IMP.Particle(m)

# "Decorate" the Particles with x,y,z attributes (point-like particles)
d1 = IMP.core.XYZ.setup_particle(p1)
d2 = IMP.core.XYZ.setup_particle(p2)

# Use some XYZ-specific functionality (set coordinates)
d1.set_coordinates(IMP.algebra.Vector3D(10.0, 10.0, 10.0))
d2.set_coordinates(IMP.algebra.Vector3D(-10.0, -10.0, -10.0))
print d1, d2
```

```

import IMP
import IMP.algebra
import IMP.core

m = IMP.Model()

# Create two "untyped" Particles
p1 = IMP.Particle(m)
p2 = IMP.Particle(m)

# "Decorate" the Particles with x,y,z attributes (point-like particles)
d1 = IMP.core.XYZ.setup_particle(p1)
d2 = IMP.core.XYZ.setup_particle(p2)

# Use some XYZ-specific functionality (set coordinates)
d1.set_coordinates(IMP.algebra.Vector3D(10.0, 10.0, 10.0))
d2.set_coordinates(IMP.algebra.Vector3D(-10.0, -10.0, -10.0))
print d1, d2

#include <fstream>

#include <IMP.h>
#include <IMP/algebra.h>
#include <IMP/core.h>

int main()
{
    IMP::Pointer<IMP::Model> m = new IMP::Model();

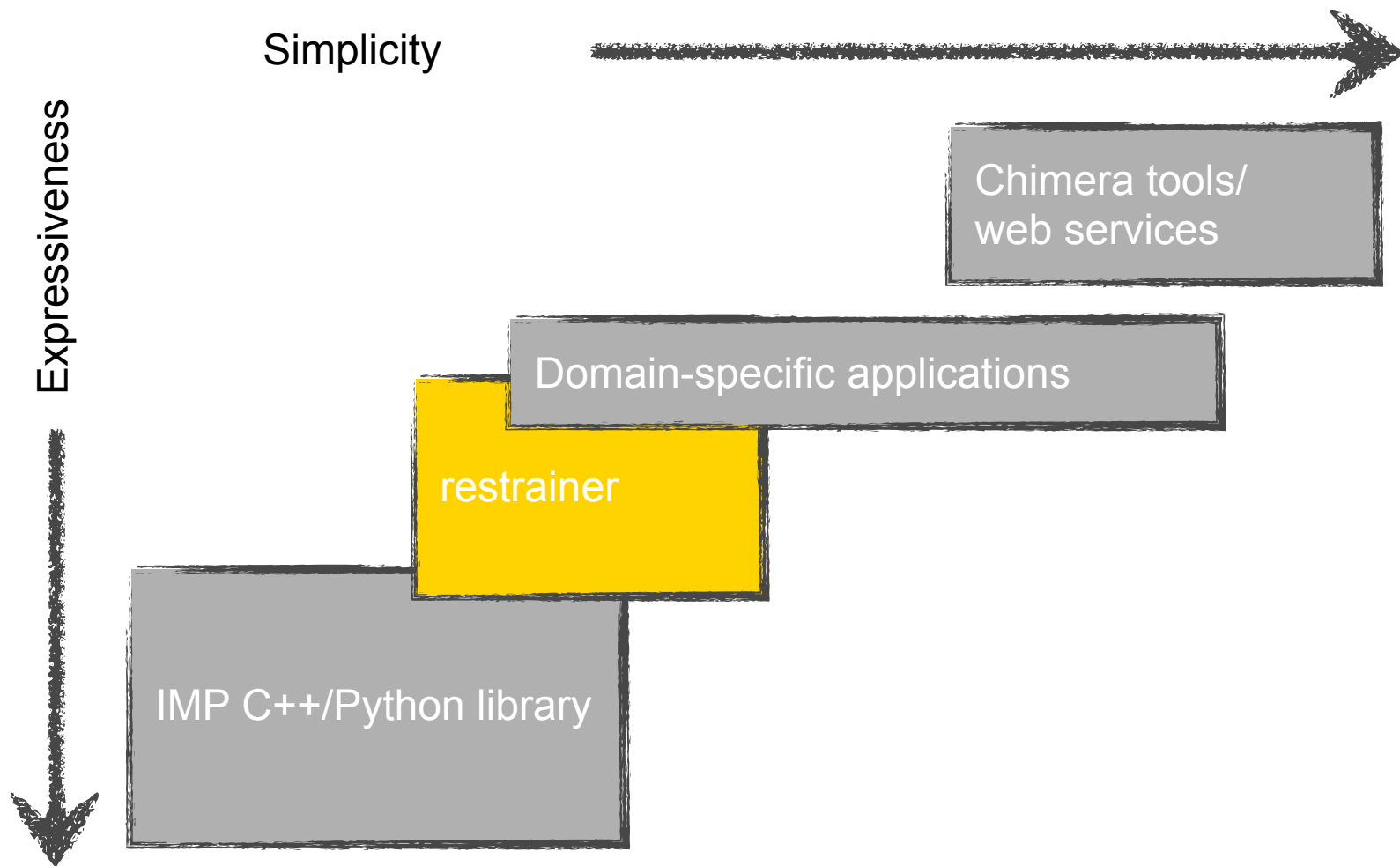
    // Create two "untyped" Particles
    IMP::Pointer<IMP::Particle> p1 = new IMP::Particle(m);
    IMP::Pointer<IMP::Particle> p2 = new IMP::Particle(m);

    // "Decorate" the Particles with x,y,z attributes (point-like particles)
    IMP::core::XYZ d1 = IMP::core::XYZ::setup_particle(p1);
    IMP::core::XYZ d2 = IMP::core::XYZ::setup_particle(p2);

    // Use some XYZ-specific functionality (set coordinates)
    d1.set_coordinates(IMP::algebra::Vector3D(10.0, 10.0, 10.0));
    d2.set_coordinates(IMP::algebra::Vector3D(-10.0, -10.0, -10.0));
    std::cout << d1 << " " << d2 << std::endl;

    return 0;
}

```

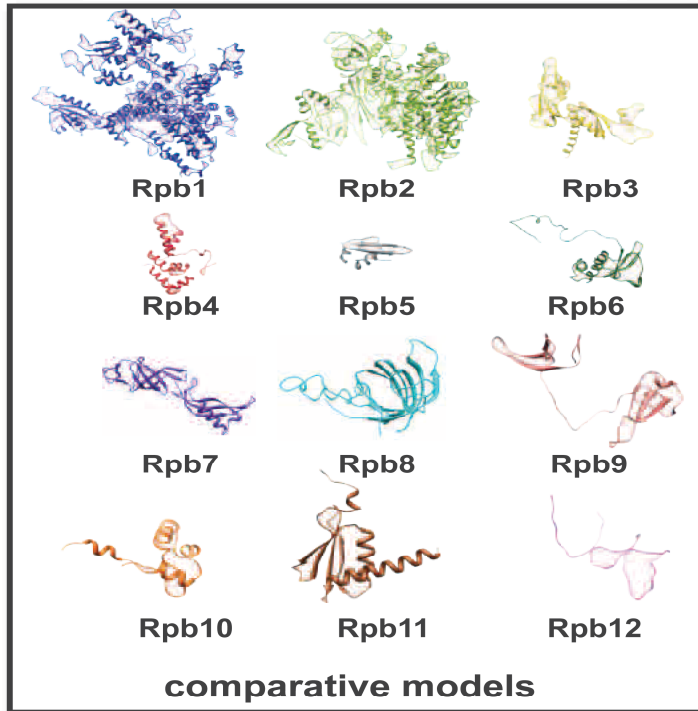


# “Restrainer” simplified interface

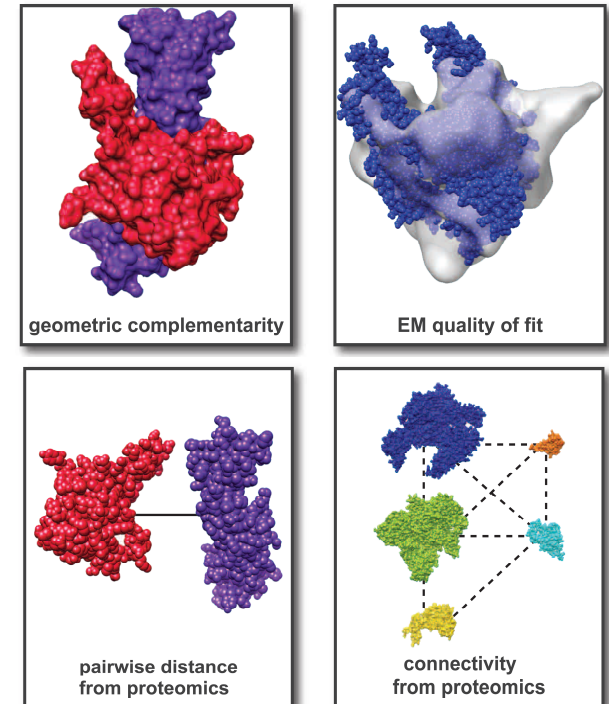
- The IMP kernel gives us lots of flexibility to set up the system and restraints exactly how we want them
- Many more classes available: see <http://salilab.org/imp/> for a comprehensive list and examples
- But: often we want a simpler interface to solve modeling problems
- Restrainer is a higher-level interface that simplifies the setup of a complex system
- Optimization, however, may still need to be tweaked for your system

# “Restrainer” simplified interface

## Representation



## Data Translation into Spatial Restraints

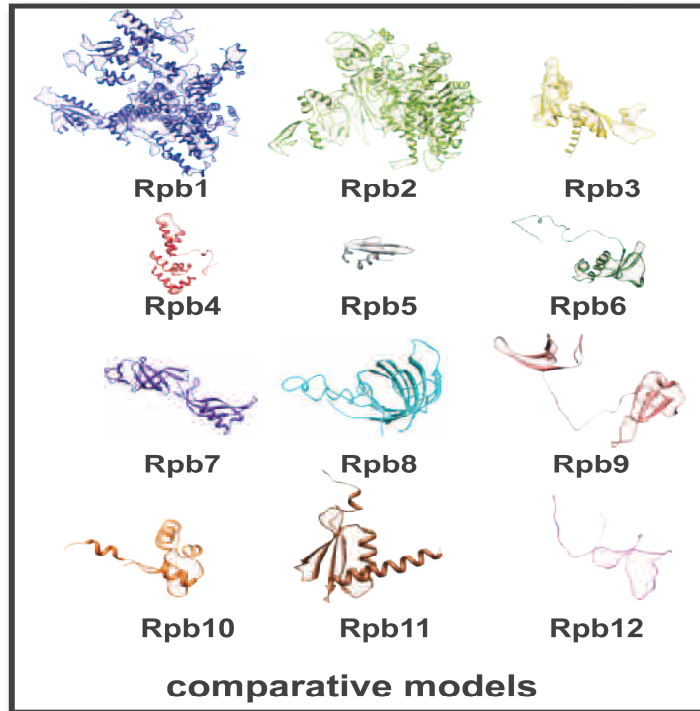


Elina Tjioe, Keren Lasker

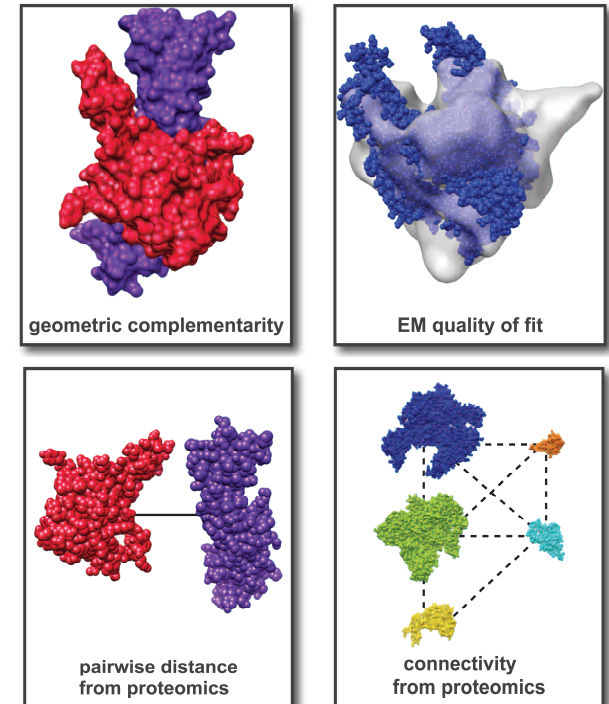


# “Restrainer” simplified interface

## Representation



## Data Translation into Spatial Restraints



```
<Representation>
```

```
<Protein id="Rpb1"><Chain filename="Rpb1.pdb"/></Protein>
```

```
.  
. .  
.
```

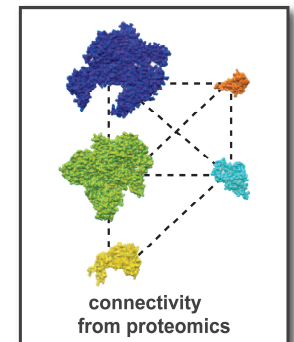
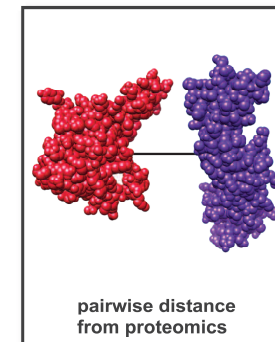
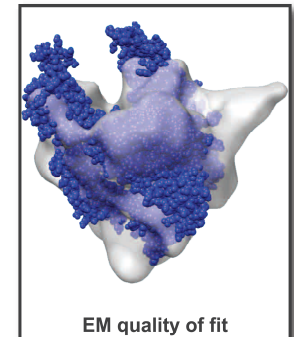
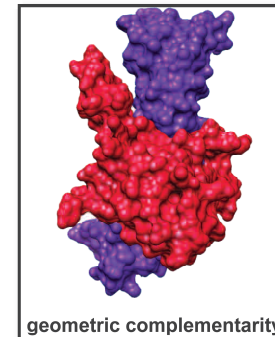
```
<Protein id="Rpb12"><Chain filename="Rpb12.pdb"/></Protein>
```

```
</Representation>
```

# “Restrainer” simplified interface

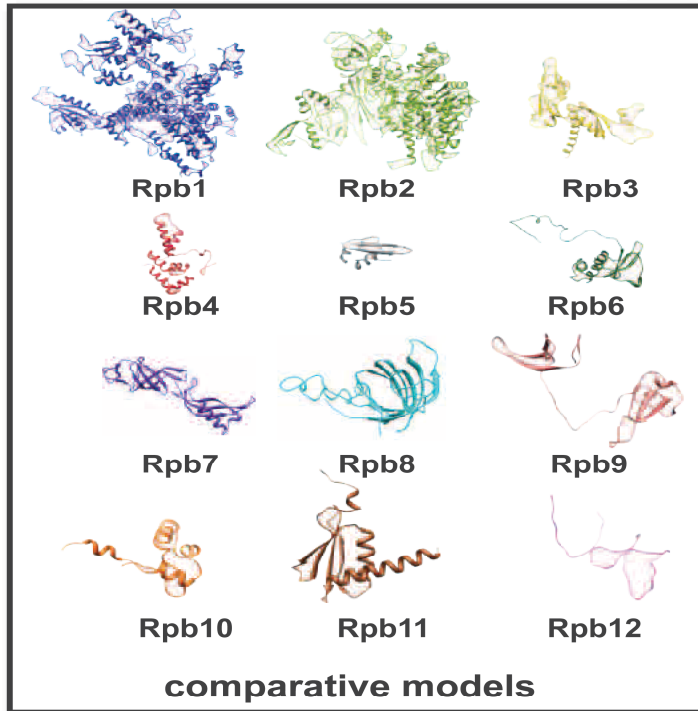
```
<RestraintSet>
  <EM>
    <Restraint density_filename="in.mrc">
      <Particle id="Rpb1"/>
    </Restraint>
  </EM>
  <Distance>
    <Restraint distance="5.0" std_dev="0.1">
      <Particle id="Rpb1"/>
      <Particle id="Rpb4"/>
    </Restraint>
  </Distance>
  <Y2H>
    <Restraint>
      <Particle id="Rpb2"/>
      <Particle id="Rpb3"/>
      <Particle id="Rpb8"/>
    </Restraint>
  </Y2H>
</RestraintSet>
```

## Data Translation into Spatial Restraints

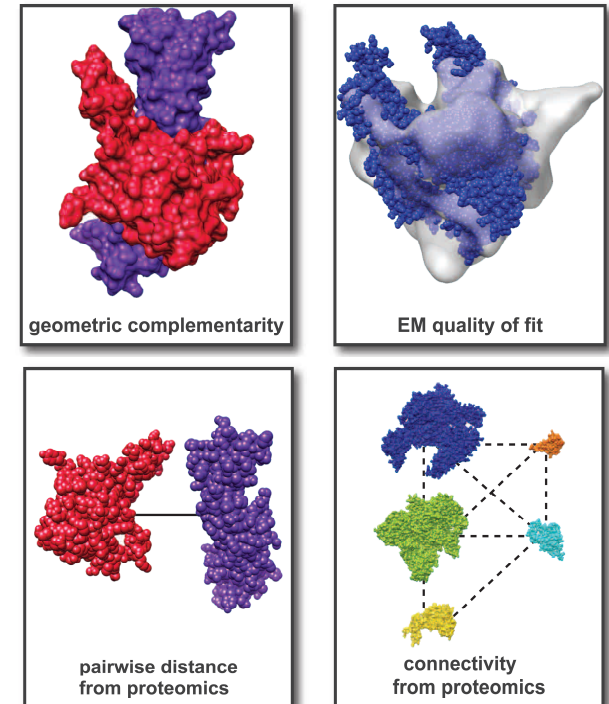


# “Restrainer” simplified interface

## Representation

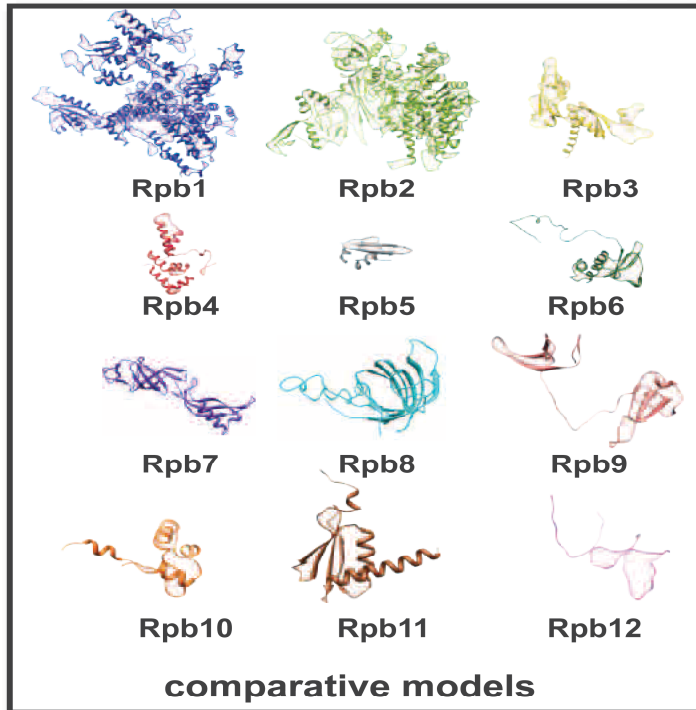


## Data Translation into Spatial Restraints

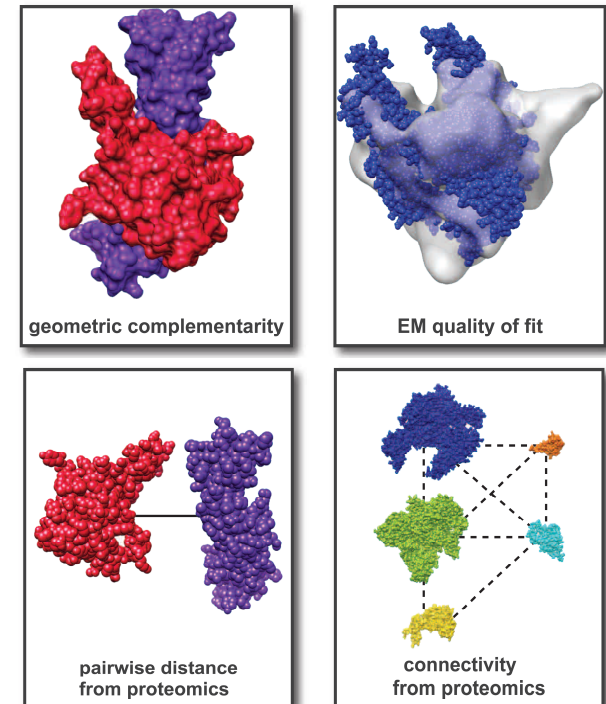


# “Restrainer” simplified interface

## Representation



## Data Translation into Spatial Restraints



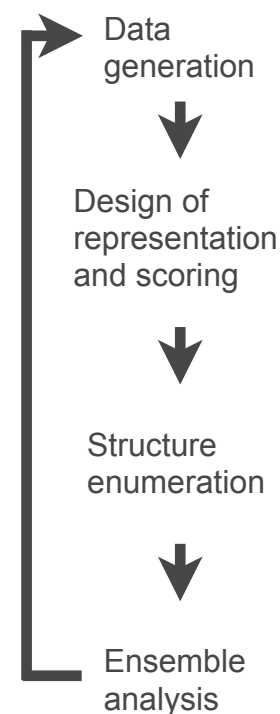
```
representation = IMP.restrainer.XMLRepresentation("repr.xml").run()  
restraint = IMP.restrainer.XMLRestraint("restraint.xml").run()
```

```
model = representation.to_model()  
restraint.add_to_representation(representation)
```

```
s = IMP.multifit.run_optimization(model)  
IMP.statistics.analyze_solution_ensemble(s)
```

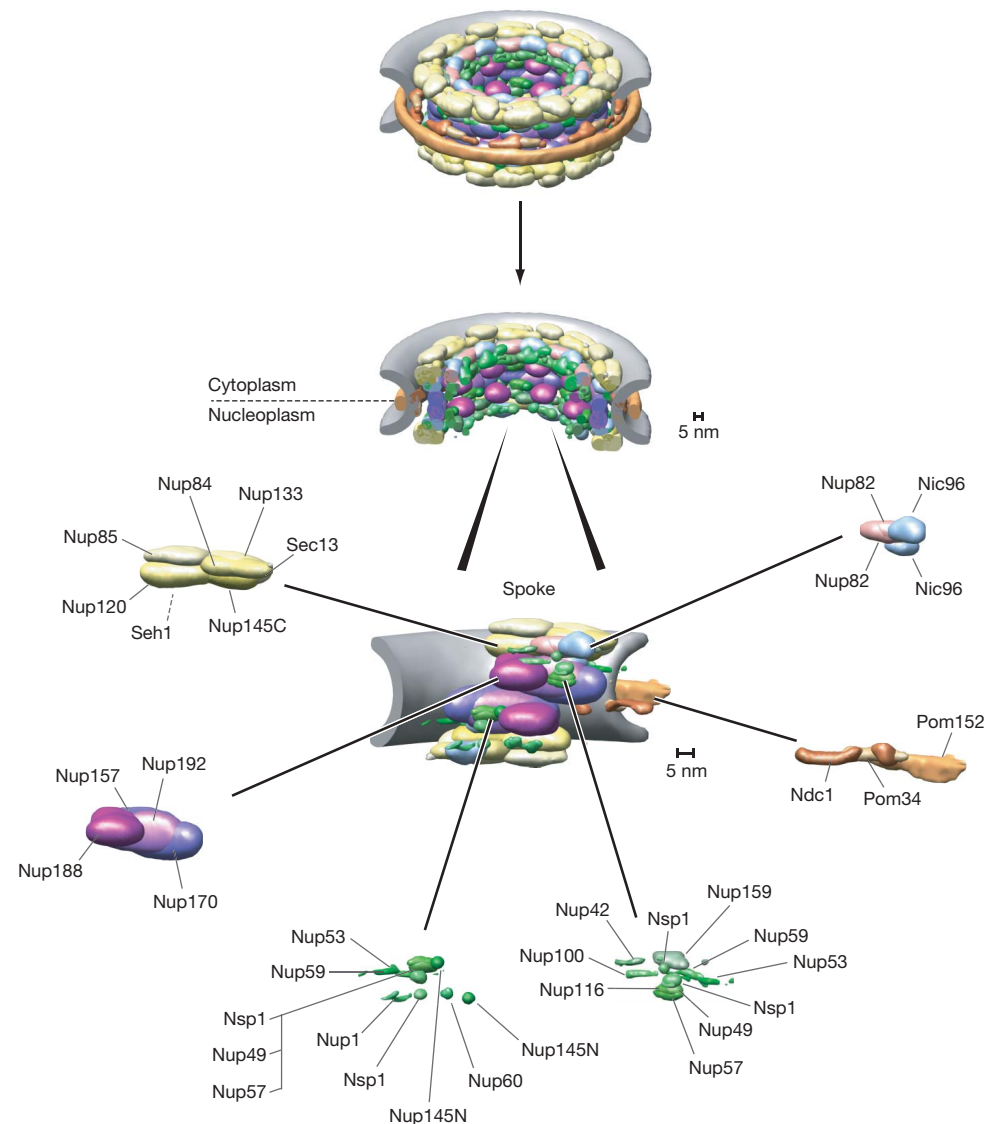
# “Restrainer” simplified interface

- Complete modeling protocols can be published as XML inputs and a Python script
- Reproducible
- Reference model from ensemble analysis
- Guide future experiments
- Identify poor experiments
- Show effect of new data



# Restrainer example

- Determine a simple “bead model” of a subcomplex of the Nuclear Pore Complex, Nup84

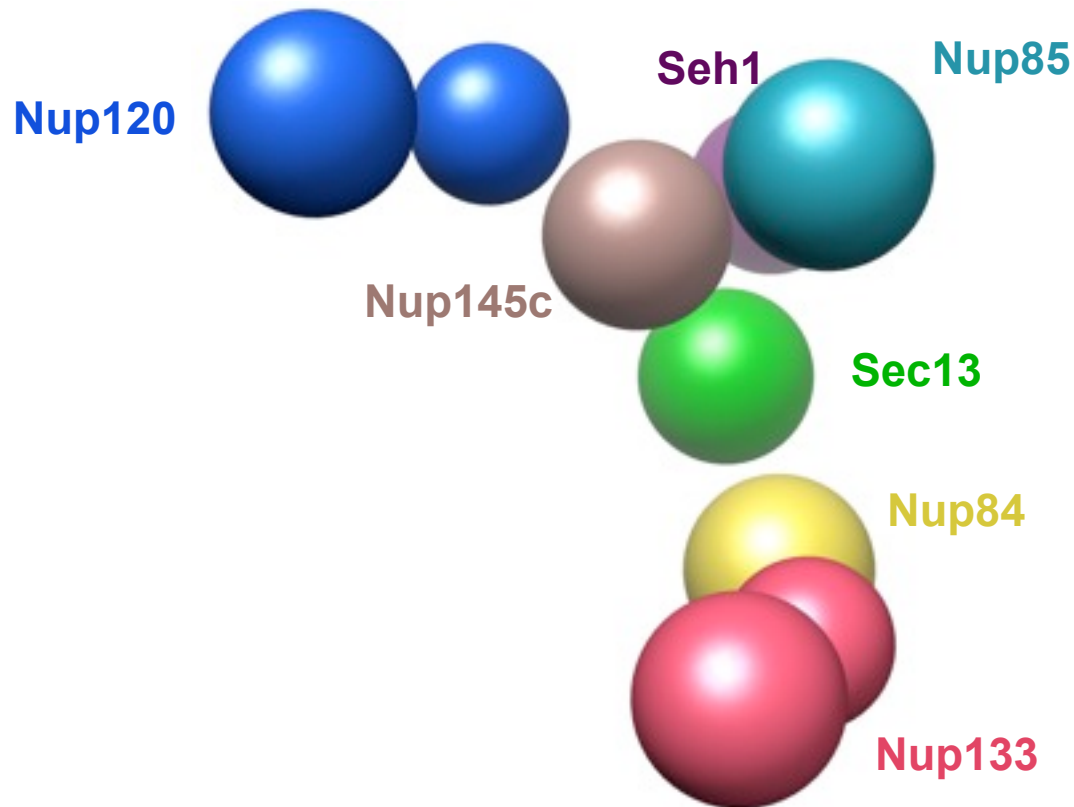


# Restrainer input files

- Nup84Complex\_bead\_repr\_1.xml
  - Defines the representation of the system
  - In this case, each protein is represented by a single sphere or (for Nup133 and Nup120) two spheres
  - Larger proteins are represented by larger spheres
- Nup84Complex\_restraint\_1.xml
  - Defines the known restraints on the system
  - Here, we prevent the spheres from interpenetrating (excluded volume) and add yeast two-hybrid information
- Nup84Complex\_display\_1.xml
  - Gives each protein a color for display purposes
- nup84.py
  - Runs the optimization and generates outputs

# Run the optimization

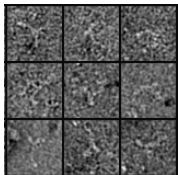
- Run like any other Python script
- Output: log file, structures (as Chimera Python inputs)



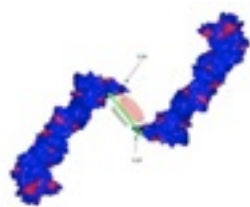


## Next steps

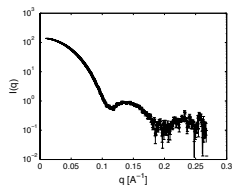
- For this model, we used only a bead representation and yeast two-hybrid (connectivity) information
- Can potentially improve the model by
  - using more beads per protein to match experimental shapes more accurately
  - using atomic structures (X-ray structures or comparative models) rather than beads
  - adding more restraints - e.g. 2D or 3D EM maps, SAXS fits, distances extracted from crystal structures of subcomplexes
  - building an ensemble of models and clustering



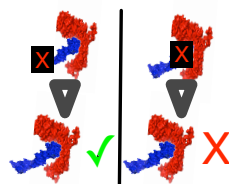
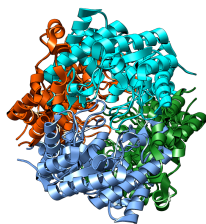
**EM density fitting**  
D. Stokes



**Chemical crosslinking**  
B. Chait, M. Rout

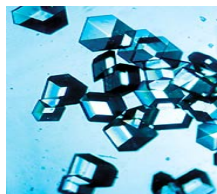


**SAXS data**  
NYSGXRC  
H. Tsuruta, SSRL

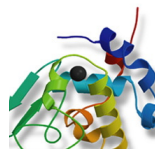


**Domain-deletion pullouts**  
M. Rout

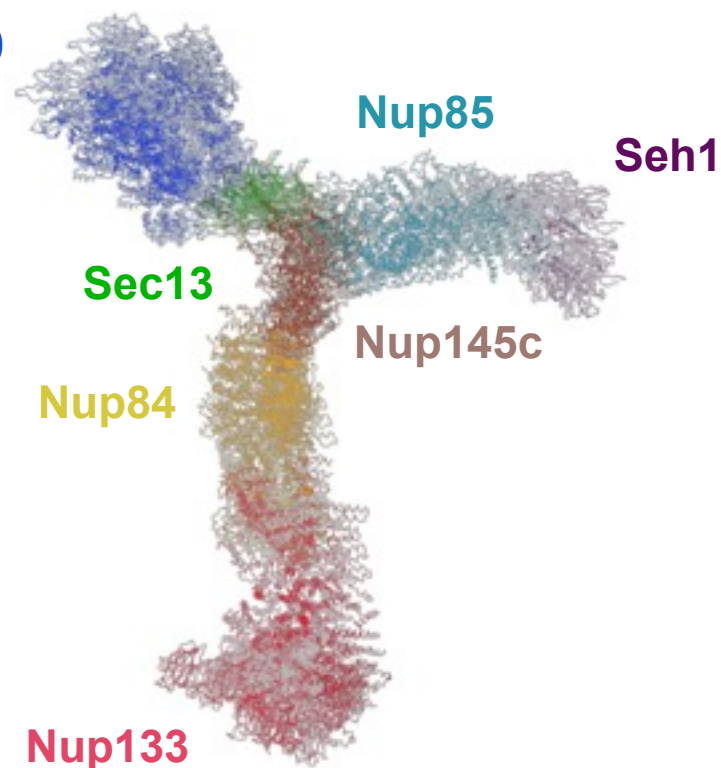
IMP  
➔

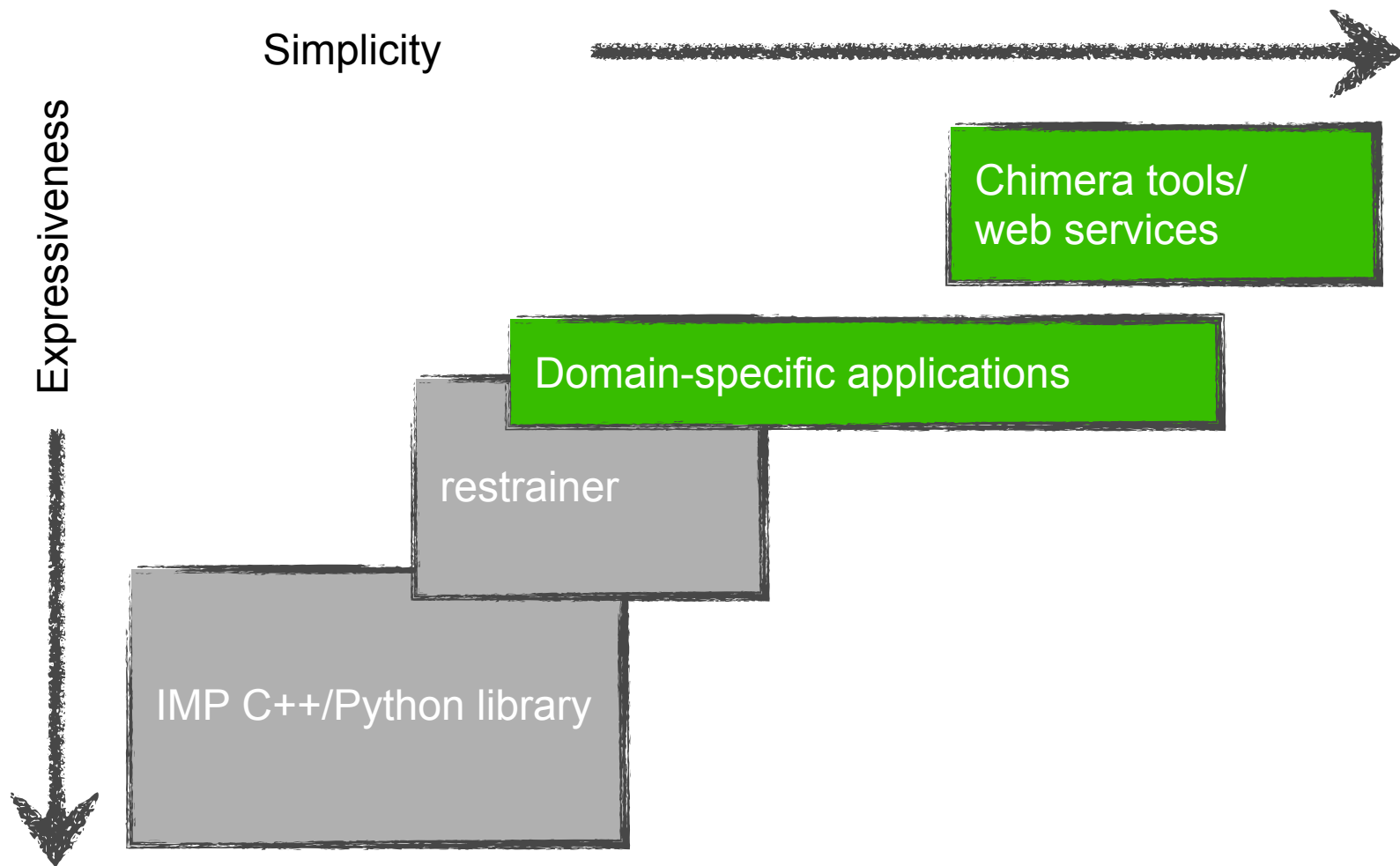


**X-ray crystallography**  
R. Stroud, CSMP  
S. Burley, NYSGXRC



**Homology modeling  
molecular docking**  
A. Sali





# Summary

- IMP provides a framework for integrative modeling
- Flexible representation, scoring, optimization
- Powerful collection of classes for developers
- Simpler XML/Python for many modeling tasks
- Web services/Chimera integration for specialized applications
- Everything freely available at <http://salilab.org/imp/>