Modeling of proteins and their assemblies with the *Integrative Modeling Platform*

Contributed by Benjamin Webb, Keren Lasker, Dina Schneidman-Duhovny, Elina Tjioe, Jeremy Phillips, Seung Joong Kim, Javier Velázquez-Muriel, Daniel Russel, and Andrej Sali, Department of Bioengineering and Therapeutic Sciences, Department of Pharmaceutical Chemistry, and California Institute for Quantitative Biosciences (QB3), University of California San Francisco, San Francisco, CA 94158, USA.

Abstract

To understand the workings of the living cell, we need to characterize protein assemblies that constitute the cell (for example, the ribosome, 26S proteasome, and the nuclear pore complex). A reliable high-resolution structural characterization of these assemblies is frequently beyond the reach of current experimental methods, such as X-ray crystallography, NMR spectroscopy, electron microscopy, footprinting, chemical cross-linking, FRET spectroscopy, small angle X-ray scattering, and proteomics. However, the information garnered from different methods can be combined and used to build computational models of the assembly structures that are consistent with all of the available datasets. Here, we describe a protocol for this integration, whereby the information is converted to a set of spatial restraints and a variety of optimization procedures can be used to generate models that satisfy the restraints as well as possible. These generated models can then potentially inform about the precision and accuracy of structure determination, the accuracy of the input datasets, and further data generation. We also demonstrate the *Integrative Modeling Platform* (IMP) software, which provides the necessary computational framework to implement this protocol, and several applications for specific use cases.

Key Words

Integrative modeling, protein structure modeling, macromolecular assemblies proteomics, X-ray crystallography, electron microscopy, SAXS.

Introduction

To understand the function of a macromolecular assembly, we must know the structure of its components and the interactions between them**(*1-4*)**. However, direct experimental determination of such a structure is generally rather difficult.

While multiple methods do exist for structure determination, each has a drawback. For example, crystals suitable for X-ray crystallography cannot always be produced, especially for large assemblies of multiple components**(*5*)**. Cryo-electron microscopy (cryo-EM), on the other hand, can be used to study large assemblies, but it is generally limited to worse than atomic resolution**(*6-8*)**. Finally, proteomics techniques, such as yeast two-hybrid**(*9*)** and mass spectrometry**(*10*)**, yield information about the interactions between proteins, but not the positions of these proteins within the assembly or the structures of the proteins themselves.

Integrative modeling

One approach to solve the structures of proteins and their assemblies is by integrative modeling, in which information from different methods is considered simultaneously during the modeling procedure. The approach is briefly outlined here for clarity; it has been covered in greater detail previously**(*11-17*)**. These methods can include experimental techniques, such as X-ray crystallography**(*5*)**, nuclear magnetic resonance (NMR) spectroscopy**(*18-20*)**, electron microscopy (EM)**(*6-8*)**, footprinting**(*21, 22*)**, chemical cross-linking**(*23-26*)**, FRET spectroscopy**(*27*)**, small angle X-ray scattering (SAXS)**(*28, 29*)**, and proteomics**(*30*)**. Theoretical sources of information about the assembly can also be incorporated, such as template structures used in comparative modeling**(*31, 32*)**, scoring functions used in molecular docking**(*33*)**, as well as other statistical preferences**(*34, 35*)** and physics-based energy functions**(*36-38*)**. Different methods yield information about different aspects of structure and at different levels of resolution. For example, atomic resolution structures may be available for individual proteins in the assembly; in other cases, only their approximate size, approximate shape, or interactions with other proteins may be known. Thus, integrative modeling techniques generate models at the resolution that is consistent with the input information. An example of a simple integrative approach is building a pseudo-atomic model of a large assembly, such as the 26S proteasome**(*39, 40*)**, by fitting atomic structures of its subunits predicted by comparative protein structure modeling into a density map determined by cryo-EM**(*41, 42*)**.

The integrative modeling procedure used here**(*12, 17*)** is shown in Figure 1. The first step in the procedure is to collect all experimental, statistical, and physical information that describes the system of interest. A suitable representation for the system is then chosen and the available information is translated to a set of spatial restraints on the components of the system. For example, in the case of characterizing the molecular architecture of the nuclear pore complex (NPC)**(*12, 13*)**, atomic structures of the protein subunits were not available, but the approximate size and shape of each protein was known, so each protein was represented as a 'string' of connected spheres consistent with the protein size and shape. A simple distance between two proteins can be restrained by a harmonic function of the distance, while the fit of a model into a 3D cryo-EM density map can be restrained by the cross-correlation between the map and the computed density of the model. Next, the spatial restraints are summed into a single scoring function that can be sampled using a variety of optimizers, such as conjugate gradients,

molecular dynamics, Monte Carlo, and inference-based methods**(*42*)**. This sampling generates an ensemble of models that are as consistent with the input information as possible. In the final step, the ensemble is analyzed to determine, for example, whether all of the restraints have been satisfied or certain subsets of data conflict with others. The analysis may generate a consensus model, such as the probability density for the location of each subunit in the assembly.

Integrative Modeling Platform

We have developed the *Integrative Modeling Platform* (IMP) software (http://salilab.org/imp/)**(*12-15*)** to implement the integrative modeling procedure described above. Integrative modeling problems vary in size and scope, and thus IMP offers a great deal of flexibility and several abstraction levels as part of a multi-tiered platform (Figure 2). At the lowest level, IMP provides building blocks and tools to allow methods developers to convert data from new experimental methods into spatial restraints, to implement optimization and analysis techniques, and to implement an integrative modeling procedure from scratch; the developer can use the C++ and Python programming languages to achieve these tasks. Higher abstraction levels, designed to be used by IMP users with no programming experience, provide less flexible but more user-friendly applications to handle specific tasks, such as fitting of proteins into a density map of their assembly, or comparing a structure with the corresponding SAXS profile. IMP is freely available as open source software under the terms of the GNU Lesser General Public License (LGPL). Integrative modeling, due to its use of multiple sources of information, is often a highly collaborative venture, and thus benefits from openness of the modeling protocols and the software itself.

Materials

To follow the examples in this discussion, both the IMP software itself and a set of suitable input files are needed. The IMP software can be downloaded from http://salilab.org/imp/download.html and is available in binary form for most common machine types and operating systems; alternatively, it can be rebuilt from the source code. The example files can be downloaded from http://salilab.org/imp/tutorials/basic.zip.

Methods

The IMP C++/Python library

The core of IMP is the C++/Python library, which provides all of the necessary components, as a set of classes and modules, to allow methods developers to build an integrative modeling protocol from scratch. Most users of IMP will use one of the higher-level interfaces described in later sections; however, we will briefly demonstrate this library here to illustrate the core IMP concepts that these interfaces rely on.

The IMP library is split into a kernel and a set of extension modules (Table 1). The kernel is a small collection of classes that define the storage of information about the system and the main interfaces used to interact with that information. The information is stored in a set of **`Particle`** objects; these are flexible and abstract data containers, able to hold whatever information is necessary to represent the system. For example, a given **`Particle`** may be assigned *x*, *y*, and *z* attributes to store point coordinates, another may be assigned *x*, *y*, *z*, and a radius to represent a sphere, and another may contain two pointers to other **`Particle`**s to represent a bond or another relationship. The kernel defines only the abstract interfaces to manipulate the data in the **`Particle`**s, but does not provide implementations; these are provided in the extension modules. For example, it merely defines a **`Restraint`** as any object that, given a set of **`Particle`**s, returns a score, and an **`Optimizer`** as an object that changes the attributes of all **`Particle`**s to yield an optimized score over all restraints. It is the **`core`** module that provides, for example, a concrete **`Restraint`** acting like a harmonic 'spring' between two point-like **`Particle`**s, an **`Optimizer`** that utilizes the conjugate gradients minimization method, and much other functionality.

IMP includes a variety of modules (Table 1). Some modules provide the basic building blocks needed to construct a protocol, such as the **`core`** module that provides functionality including harmonic restraints, point-like and spherical particles, and basic optimizers, and the **`atom`** module that provides atom-like particles, a molecular dynamics optimizer, *etc*. Other modules provide support for specific types of experimental data or specialized optimizers, such as the **`em`** module that supports electron microscopy data, and the **`domino`** module that provides an inference-based divide-and-conquer optimizer. IMP is designed so that it is easy to add a new module; for example, a developer working on incorporating data from a new experimental technique may add a new IMP module that translates the data from this technique into spatial restraints.

IMP is primarily implemented in C++ for speed; however, each of the classes is wrapped so that it can also be used from Python. A protocol can thus be developed from scratch by simply writing a Python script. As an example, we will first look at the script **`simple.py`** in the '**`library`**' subdirectory of the zipfile downloaded above (Figure 3).

In the first part of the script, the IMP kernel and the **`algebra`** and **`core`** modules are loaded, as regular Python modules. We then proceed to set up the representation of the system, using the **`Model`** and **`Particle`** classes defined in the kernel. The **`Model`** class represents the entire system, and keeps track of all the **`Particle`**s, **`Restraint`**s, and links between them. As mentioned earlier, the **`Particle`** class is a flexible container, but here we give the two **`Particle`**s (p1 and p2) point-like attributes using the **`XYZ`** class defined in the **`core`** module. This **`XYZ`** class is known as a 'decorator'; it does not create a new **`Particle`**, but merely

presents a new interface to an existing **Particle**, in this case a point-like one. (Multiple decorators can be applied to a single **Particle**; for example, an atom-like **Particle** could be treated like a point, a sphere, an electrically charged particle, or an atom.) We can then treat each **Particle** like a point using methods in the **XYZ** class, here setting the *x*, *y*, and *z* coordinates to a provided vector.

In the second part, we set up the scoring of the system. We add two restraints to the **Model**, one of which harmonically restrains p1 to the origin and the other of which restrains p1 and p2 to be distance 5.0 apart. (IMP does not enforce any units of distance; however, some physical optimizers, such as molecular dynamics, expect distances to be in angstroms.) Note that the **core** module provides suitable building block restraints for this purpose. In the first case, we use the **SingletonRestraint** class that creates a restraint on a single particle (p1). It delegates the task of actually scoring the particle, however, to another class called **SingletonScore** that is simply given the **Particle** and asked for its score. In this example, we use a type of **SingletonScore** called a **DistanceToSingletonScore** that calculates the Cartesian distance between the point-like **Particle** and a fixed point (in this case the origin), and again delegates the task of scoring the distance to another class, a **UnaryFunction**. In this case, the **UnaryFunction** is a simple harmonic function with a mean of zero. Thus, the **Particle** p1 is harmonically restrained to be at the origin. The second restraint is set up similarly; however, in this case the restraints and scores act on a pair of particles. This building block functionality makes it easy to add a new type of restraint; for example, to implement a van der Waals potential it is only necessary to provide a suitable **PairScore** that scores a single pair of particles; the functionality for efficiently enumerating all pairs of such particles is already provided in IMP.

Finally, in the third part of the script, we tell IMP that it can move the two point-like particles, and to build a system configuration that is consistent with all the restraints. In this example, a simple conjugate gradients optimization is used.

The script is a regular Python script. Thus, provided that both IMP and Python are installed, it can be run on any machine, by typing on a command line, in the same directory as the script:

```
python simple.py
```

The script will run the optimization, printing IMP log messages as it goes, and finally print the coordinates of the optimized particles.

IMP is designed such that the C++ and Python interfaces are similar to use. Thus, IMP applications or protocols can be constructed either in C++ or in Python, and new IMP functionality (for example, new types of **Restraint**) can be implemented

in either language. For a comparison, please inspect the **simple.cpp** file. This file implements the same protocol as the first part of **simple.py** but uses the IMP C++ classes rather than their Python equivalents. The two programs are very similar; the only differences are in the language syntax (*eg*, the Python '`import IMP`' translates to '`#include <IMP.h>`' in C++) and in memory handling (Python handles memory automatically; in C++, memory handling must be done explicitly by using the **IMP::Pointer** class, which adds reference counting to automatically clean up after IMP objects when they are not used anymore).

restrainer: a high-level interface for integrative modeling

The IMP C++/Python library offers a great deal of flexibility in setting up the system and restraints. However, in many cases, a simpler interface to solve modeling problems is preferable. The **restrainer** IMP module is one such interface that simplifies the set up of a complex system, generating the system representation and restraints from a pair of XML files. Optimization, however, may still need to be adjusted for specific cases.

As a simple demonstration of the module, we consider the construction of a model of a subcomplex of the NPC**(*12, 13*)**. The yeast NPC is a large assembly of 50 MDa containing 456 proteins of 30 different types. The modeling of the entire assembly is beyond the scope of this tutorial; however, it has been observed that the NPC is made up of a set of smaller subcomplexes (Figure 4). One of these complexes is the Nup84 complex, consisting of seven proteins, and the modeling of this complex is illustrated in this tutorial.

All of the XML and Python files necessary to perform the Nup84 modeling can be found in the '**restrainer**' subdirectory of the zipfile downloaded above. The first of these XML files is **representation.xml**, which determines how the system is represented. IMP does not require every protein in the system to be modeled with the same representation; for example, some proteins could be modeled as sets of atoms and others at a lower resolution. As for the original NPC modeling, here we use a 'bead model' for the Nup84 complex; each protein is represented as a sphere, or a pair of spheres (in the case of the more rodlike Nup133 and Nup120 proteins), with larger proteins using larger spheres. The second XML file encodes the input structural data as spatial restraints on the system. Here, we use two simple sources of information. First, excluded volume for each protein. Second, yeast two-hybrid results for some pairs of proteins. The third XML file is for visualization only, and assigns each sphere a different color. Finally, the Python script loads in all three of the XML files and performs a simple conjugate gradients optimization. This Python script can be executed just like any other Python script:

```
python nup84.py
```

restrainer first generates a set of sphere-like particles to represent the system. It then converts the information in the restraints file into a set of IMP restraints. It

generates an excluded volume restraint that prevents each protein sphere from penetrating any other sphere and a set of 'connectivity' restraints**(*12*)** that force the protein particles to reproduce the interactions implied by the yeast two-hybrid experiments. The optimization generates a file **`optimized.py`** that is an input file for the molecular visualization program Chimera**(*43*)**; when loaded into Chimera, it displays the final optimized configuration of the complex (Figure 5).

In this example, the modeling problem is simple and thus generating a single model is sufficient to find a solution that satisfies all restraints. However, when all such models need to be found or, in more complex cases, when a global solution of the scoring function is hard to find (for example, because restraints are contradictory due to errors in experiments or experiment interpretations), the modeling procedure is repeated to generate an ensemble of models. When modeling the NPC, the top-scoring models were clustered and used to generate a probability density for each component within the complex**(*12*)**. The envelope of this density defined the precision of the corresponding component localization. Only a single cluster of structures was found that satisfied all of the restraints. If contradictory information is presented, however, the optimization will be frustrated, unable to find solutions that simultaneously satisfy all restraints. The ensemble of solutions will exhibit more variability than that in a non-frustrated case. Such frustration can be tested for in the iterative integrative modeling procedure by removing potentially conflicting restraints and repeating the modeling. Finally, the accuracy of the generated model(s) can be gauged by comparison with experimental data that were not used in the original modeling. For example, the generated bead model of the Nup84 complex has a characteristic Y-shape, which is consistent with electron micrographs of the complex**(*44*)**, even though these data were not used in our example.

The **`restrainer`** XML and Python files, together with the experimental data, such as cryo-EM maps, constitute a complete modeling protocol. Thus, an assembly model built using this protocol can be published along with the input files to allow the model to be reproduced and easily updated. Such a model can thus act as a reference for future studies; for example, regions of the model that were poorly resolved can be investigated with new experiments, the resulting data incorporated into the protocol, and new models generated. Alternatively, existing unused experimental data can be added to the protocol to determine whether unused data is consistent with that used to build the model. The iterative nature of the protocol thus extends beyond the generation of the first 'correct' model.

Integration of comparative modeling, X-ray crystallography, and SAXS

The Nup84 complex structure determined above is consistent with all input information, but for a detailed understanding of its function, an accurate atomic structure is required. Two possible routes to such a structure, depending on the available information, are (i) fitting atomic structures of the individual protein subunits into a cryo-EM map of the assembly and (ii) accurately placing pairs of subunits relative to each other using X-ray crystallography or molecular docking.

For both routes, atomic structures of the subunits are required; these structures can be obtained *via* X-ray crystallography or comparative modeling.

One component of the Nup84 complex is the Nup133 protein; the structure of this protein has been characterized by both X-ray crystallography and SAXS**(*45*)**. SAXS differs from X-ray crystallography in that it is applied to proteins in solution rather than crystals; thus, it can be applied to a much wider range of proteins in states more closely resembling their functional forms than X-ray crystallography, but the information is rotationally averaged and so the resulting SAXS profile gives less structural information**(*29, 46, 47*)**. IMP contains a method that, given an atomic protein structure, can calculate its SAXS profile using the Debye formula, and then fit this profile against the experimentally determined one**(*48, 49*)**. This method is implemented in the IMP **`saxs`** module and so can be used by writing a suitable Python script. However, because fitting against a SAXS profile is a common task, we provide an IMP application, FoXS, which automates this process. FoXS is available both as a command-line IMP application and a web service at http://salilab.org/foxs.

All input files for this demonstration are available in the '**`saxs`**' subdirectory of the downloaded zipfile. The structure of the C-terminal domain of yeast Nup133 is available in the RCSB Protein Data Bank (PDB)**(*50*)** as code 3kfo (file **`3KFO.pdb`**), while the experimental SAXS profile is given in the **`23922_merge.dat`** file. The atomic structure can be fit against the SAXS profile by running FoXS in the directory containing both files:

```
foxs 3KFO.pdb 23922_merge.dat
```

Alternatively, the two files can be submitted to the FoXS web server. FoXS compares the theoretical profile of the provided structure (solid line in Figure 6) with the experimental profile (points), and calculates the quality of the fit, χ, with smaller values corresponding to closer fits.

The fit in this example is not a good one (χ=2.96). To understand why this is so, we examine the header of the 3kfo PDB file, which reveals two problems. Several residues at the N and C termini were not resolved in the X-ray experiment (8 in total, 2 at the N terminus and 6 at the C terminus), and the sidechains of 16 other residues could also not be located (REMARK 465 and REMARK 470 lines).

The missing 8 residues and 16 sidechains need to be placed to create a complete atomic structure. One way to achieve this goal is to build a comparative model using a package such as MODELLER (http://salilab.org/modeller/)**(*31, 32*)** relying on the original 3kfo structure as a template and the full sequence (including the 8 missing N and C terminal residues) as the target. The corresponding MODELLER alignment file (**`3KFO-fill.ali`**) and script file (**`fill.py`**) are provided in the downloaded zipfile. Each candidate comparative model can be fitted against the SAXS profile

using the FoXS command-line application or the web service in exactly the same way as the original 3kfo structure; the best MODELLER model gives a significantly improved fit between the theoretical and experimental profiles (dashed line in Figure 6; χ=1.21).

Given similar atomic structures of the subunits in the Nup84 complex, as either crystal structures or comparative models, **`restrainer`** can be used to build an atomic model of the complex. Note, however, that an accurate model of such a complex would require additional information beyond the proteomics data used above, since yeast two-hybrid data only show that proteins interact, not the specific residues in the protein-protein interaction, and thus do not inform us about the relative orientations of the interacting proteins. Such information can be obtained, for example, from chemical-crosslinking, molecular docking, or cryo-EM maps, as illustrated in the next section.

Determining macromolecular assembly structures by fitting multiple structures into an electron density map

Often, we have available high-resolution (atomic) information for the subunits in an assembly, and low-resolution information for the assembly as a whole (a cryo-EM electron density map). A high-resolution model of the whole assembly can thus be constructed by simultaneously fitting the subunits into the density map. Fitting of a single protein into a density map is usually done by calculating the electron density of the protein followed by a search of the protein position in the cryo-EM map that maximizes the cross correlation of the two maps. Simultaneously fitting multiple proteins into a given map is significantly more difficult, since an incorrect fit of one protein will also prevent other proteins from being placed correctly.

IMP contains a **`multifit(`*`41,42`*`)`** module (http://salilab.org/multifit/) that can efficiently solve such multiple fitting problems for density map resolutions as low as 25Å, relying on a general inferential optimizer DOMINO. The fitting protocol is a multi-step procedure that proceeds *via* discretization of both the map and the proteins, local fitting of the proteins into the map, and an efficient combination of local fits into global solutions (Figure 7). Here, we will demonstrate the use of **`multifit`** in building a model of the ARP2/3 complex**(*51*)** using crystal structures of its seven constituent proteins (ARP2, ARP3, and ARC1-5) and a 20Å density map of the assembly. All input files for this procedure can be found in the '**`multifit`**' subdirectory of the downloaded zipfile.

The first step in using **`multifit`** is to create input files that guide the protocol. The first of these files, **`assembly.input`**, lists each of the subunits and the density map, complete with the names of the files from which the input structures and map will be read, and those to which outputs from later steps will be written. In this case, we also know the native structure of the assembly (PDB code 1tyq) and so we add the subunit structures in native conformation to this input file (rightmost column); **`multifit`** will use them to assess its accuracy. Normally, of course, the real native

structure is not known, in which case this column in the input file is left blank. The second file, **multifit.par**, specifies various optimization parameters, and is described in more detail on the **multifit** website (http://salilab.org/multifit/).

The second step is to determine a reduced representation for both the density map and the subunits, using the Gaussian Mixture Model. This task can be achieved by typing, in the directory containing **assembly.input** (the syntax for running Python scripts may vary depending on where the files are installed):

```
/opt/multifit/utils/run_anchor_points_detection.py
assembly.input 700
```

This run determines a reduced representation of the EM map that best reproduces the configuration of all voxels with density above 700, and a similar reduced representation of each subunit as a set of 3D Gaussian functions. The number of Gaussians is specified in **assembly.input** for each subunit. It should be at least 3 (the minimum required for fitting) and each Gaussian should cover approximately the same number of residues (for example, if you choose 50 residues per Gaussian, a 170-residue protein should use 3 Gaussians and a 260-residue protein should use 5 Gaussians). Each such reduced map representation can also be thought of as an anchor point graph, where each anchor point corresponds to the center of a 3D Gaussian, and the edges in the graph correspond to the connectivity between regions of the map or protein. These reduced representations are written out as PDB files containing fake Cα atoms, where each Cα corresponds to a single anchor point.

The third step is to fit each protein in the vicinity of the EM map's anchor points. This task is achieved by running:

```
/opt/multifit/utils/run_protein_fitting.py assembly.input
multifit.par
```

The output is a set of candidate fits, where the subunit is rigidly rotated and translated to fit into the density map. Each fit is written as a PDB file in the '**fits**' subdirectory. The fitting procedure is performed by either aligning a reduced representation of a protein to a reduced representation of the density map**(*41*)** or by fitting the protein principal components to the principal components of a segmented region of the map.

Finally, the fits are scored and then combined into a set of the best-scoring global configurations:

```
/opt/multifit/utils/run_all_scores.py assembly.input >
scores.log
```

```
/opt/multifit/utils/run_multifit.py assembly.input
assembly.jt assembly_configurations.output
data/models/1tyq.fitted.pdb > multifit.log
```

The scoring function used to assess each fit includes the quality-of-fit of each subunit into the map, the protrusion of each subunit out of the map envelope, and the shape complementarity between pairs of neighboring subunits. The optimization avoids exhaustive enumeration of all possible mappings of subunits to anchor points by means of a branch-and-bound algorithm combined with the DOMINO divide-and-conquer message-passing optimizer using a discrete sampling space**(*42*)**.

The final output from **`multifit`** is a file **`assembly_configurations.output`** that lists the best global solutions, ranked by their score, an excerpt of which is shown below:

```
ARP3,0|ARP2,14|ARC1,3|ARC2,24|ARC3,19|ARC4,11|ARC5,13|(17.5
593729019)(rmsd:29.2637996674)(conf.0.pdb)
ARP3,5|ARP2,13|ARC1,9|ARC2,24|ARC3,19|ARC4,4|ARC5,13|(18.32
58602619)(rmsd:11.997220993)(conf.1.pdb)
```

For each global solution, **`multifit`** lists the indices of the local fits for each subunit and the score. Each solution is also written out as a multi-chain PDB file of the assembly. In addition, because we also provided the native structure (**`1tyq.fitted.pdb`**), the RMSD between the native conformation and each solution is listed. In this case, the RMSD measure indicates that **`multifit`** has correctly determined the architecture of the assembly, placing each subunit in the approximately correct part of the map. However, the protein-protein interfaces are clearly not accurate at the atomic level. These models could thus be refined with a combination of pairwise computational docking and local sampling, ideally supported by additional experimental data, such as chemical cross-linking, various kinds of footprinting, and X-ray crystallography of binary subunit complexes.

Summary

The structures of protein assemblies can typically not be fully characterized with any individual computational or experimental method. Integrative modeling aims to solve this problem by combining information from multiple methods to generate structural models. Integrative modeling problems can be tackled using the method of satisfaction of spatial restraints. In this approach, a suitable representation for the system is chosen, the information is converted into a set of spatial restraints, the restraints are simultaneously satisfied as well as possible by optimizing a function that is the sum of all restraints, and the resulting models are analyzed. Further experiments as well as the precision and likely accuracy of both the model and the data can be informed. IMP is an open source and flexible software package that

provides all of the components needed to implement an integrative modeling protocol from scratch. It also contains higher-level applications and web services that can tackle specific use cases more conveniently.

Acknowledgements

We are grateful to all members of our research group, especially to Frank Alber, Friedrich Förster, and Bret Peterson who contributed to early versions of IMP. We also acknowledge support from National Institutes of Health (R01 GM54762, U54 RR022220, PN2 EY016525, and R01 GM083960) as well as computing hardware support from Ron Conway, Mike Homer, Hewlett-Packard, NetApp, IBM, and Intel.

References

1. Schmeing TM, and Ramakrishnan V (2009) What recent ribosome structures have revealed about the mechanism of translation, *Nature 461*, 1234-1242.
2. Sali A, Glaeser R, Earnest T, and Baumeister W (2003) From words to literature in structural proteomics, *Nature 422*, 216-225.
3. Mitra K, and Frank J (2006) Ribosome dynamics: insights from atomic structure modeling into cryo-electron microscopy maps, *Annu Rev Biophys Biomol Struct 35*, 299-317.
4. Robinson C, Sali A, and Baumeister W (2007) The molecular sociology of the cell, *Nature 450*, 973-982.
5. Blundell T, and Johnson L (1976) *Protein Crystallography*, Academic Press, New York.
6. Stahlberg H, and Walz T (2008) Molecular electron microscopy: state of the art and current challenges, *ACS Chem Biol 3*, 268-281.
7. Chiu W, Baker ML, Jiang W, Dougherty M, and Schmid MF (2005) Electron cryomicroscopy of biological machines at subnanometer resolution, *Structure 13*, 363-372.
8. Lucic V, Leis A, and Baumeister W (2008) Cryo-electron tomography of cells: connecting structure and function, *Histochem Cell Biol 130*, 185-196.
9. Parrish JR, Gulyas KD, and Finley RL Jr. (2006) Yeast two-hybrid contributions to interactome mapping, *Curr Opin Biotechnol 17*, 387-393.
10. Gingras AC, Gstaiger M, Raught B, and Aebersold R (2007) Analysis of protein complexes using mass spectrometry, *Nat Rev Mol Cell Biol 8*, 645-654.
11. Alber F, Kim M, and Sali A (2005) Structural characterization of assemblies from overall shape and subcomplex compositions, *Structure 13*, 435-445.
12. Alber F, Dokudovskaya S, Veenhoff L, Zhang W, Kipper J, Devos D, Suprapto A, Karni-Schmidt O, Williams R, Chait B, Rout M, and Sali A (2007) Determining the architectures of macromolecular assemblies, *Nature 450*, 683-694.
13. Alber F, Dokudovskaya S, Veenhoff L, Zhang W, Kipper J, Devos D, Suprapto A, Karni-Schmidt O, Williams R, Chait B, Sali A, and Rout M (2007) The molecular architecture of the nuclear pore complex, *Nature 450*, 695-701.

14. Lasker K, Phillips JL, Russel D, Velazquez-Muriel J, Schneidman-Duhovny D, Webb B, Schlessinger A, and Sali A (2010) Integrative Structure Modeling of Macromolecular Assemblies from Proteomics Data, *Mol Cell Proteomics, epub ahead of print*.
15. Russel D, Lasker K, Phillips J, Schneidman-Duhovny D, Velazquez-Muriel J, and Sali A (2009) The structural dynamics of macromolecular processes, *Curr Opin Cell Biol 21*, 97-108.
16. Alber F, Forster F, Korkin D, Topf M, and Sali A (2008) Integrating diverse data for structure determination of macromolecular assemblies, *Annu Rev Biochem 77*, 443-477.
17. Alber F, Chait BT, Rout MP, and Sali A (2008) Integrative Structure Determination of Protein Assemblies by Satisfaction of Spatial Restraints, In *Protein-protein interactions and networks: identification, characterization and prediction.* (Panchenko, A., and Przytycka, T., Eds.), pp 99-114, Springer-Verlag, London, UK.
18. Bonvin AM, Boelens R, and Kaptein R (2005) NMR analysis of protein interactions, *Curr Opin Chem Biol 9*, 501-508.
19. Fiaux J, Bertelsen EB, Horwich AL, and Wuthrich K (2002) NMR analysis of a 900K GroEL GroES complex, *Nature 418*, 207-211.
20. Neudecker P, Lundstrom P, and Kay LE (2009) Relaxation dispersion NMR spectroscopy as a tool for detailed studies of protein folding, *Biophys J 96*, 2045-2054.
21. Takamoto K, and Chance MR (2006) Radiolytic protein footprinting with mass spectrometry to probe the structure of macromolecular complexes, *Annu Rev Biophys Biomol Struct 35*, 251-276.
22. Guan JQ, and Chance MR (2005) Structural proteomics of macromolecular assemblies using oxidative footprinting and mass spectrometry, *Trends Biochem Sci 30*, 583-592.
23. Taverner T, Hernandez H, Sharon M, Ruotolo BT, Matak-Vinkovic D, Devos D, Russell RB, and Robinson CV (2008) Subunit architecture of intact protein complexes from mass spectrometry and homology modeling, *Acc Chem Res 41*, 617-627.
24. Chen ZA, Jawhari A, Fischer L, Buchen C, Tahir S, Kamenski T, Rasmussen M, Lariviere L, Bukowski-Wills JC, Nilges M, Cramer P, and Rappsilber J (2010) Architecture of the RNA polymerase II-TFIIF complex revealed by cross-linking and mass spectrometry, *EMBO J 29*, 717-726.
25. Sinz A (2006) Chemical cross-linking and mass spectrometry to map three-dimensional protein structures and protein-protein interactions, *Mass Spectrom Rev 25*, 663-682.
26. Trester-Zedlitz M, Kamada K, Burley SK, Fenyo D, Chait BT, and Muir TW (2003) A modular cross-linking approach for exploring protein interactions, *J Am Chem Soc 125*, 2416-2425.
27. Joo C, Balci H, Ishitsuka Y, Buranachai C, and Ha T (2008) Advances in single-molecule fluorescence methods for molecular biology, *Annu Rev Biochem 77*, 51-76.

28. Mertens HD, and Svergun DI (2010) Structural characterization of proteins and complexes using small-angle X-ray solution scattering, *J Struct Biol*.
29. Hura GL, Menon AL, Hammel M, Rambo RP, Poole FL 2nd, Tsutakawa SE, Jenney FE Jr, Classen S, Frankel KA, Hopkins RC, Yang SJ, Scott JW, Dillard BD, Adams MW, and Tainer JA (2009) Robust, high-throughput solution structural analyses by small angle X-ray scattering (SAXS), *Nat Methods 6*, 606-612.
30. Berggard T, Linse S, and James P (2007) Methods for the detection and analysis of protein-protein interactions, *Proteomics 7*, 2833-2842.
31. Sali A, and Blundell TL (1993) Comparative protein modelling by satisfaction of spatial restraints, *J Mol Biol 234*, 779-815.
32. Sali A, and Blundell TL (1994) Comparative protein modeling by statisfaction of spatial restraints, In *Protein Structure by Distance Analysis* (Bohr, H., and Brunak, S., Eds.), pp 64-86, TECH UNIV DENMARK, CTR BIOL SEQUENCE ANAL, LYNGBY, DENMARK.
33. Vajda S, and Kozakov D (2009) Convergence and combination of methods in protein-protein docking, *Curr Opin Struct Biol 19*, 164-170.
34. Shen MY, and Sali A (2006) Statistical potential for assessment and prediction of protein structures, *Protein Sci 15*, 2507-2524.
35. Melo F, Sanchez R, and Sali A (2002) Statistical potentials for fold assessment, *Protein Sci 11*, 430-448.
36. Brooks BR, Brooks CL 3rd, Mackerell AD Jr, Nilsson L, Petrella RJ, Roux B, Won Y, Archontis G, Bartels C, Boresch S, Caflisch A, Caves L, Cui Q, Dinner AR, Feig M, Fischer S, Gao J, Hodoscek M, Im W, Kuczera K, Lazaridis T, Ma J, Ovchinnikov V, Paci E, Pastor RW, Post CB, Pu JZ, Schaefer M, Tidor B, Venable RM, Woodcock HL, Wu X, Yang W, York DM, and Karplus M (2009) CHARMM: the biomolecular simulation program, *J Comput Chem 30*, 1545-1614.
37. Case DA, Cheatham TE 3rd, Darden T, Gohlke H, Luo R, Merz KM Jr, Onufriev A, Simmerling C, Wang B, and Woods RJ (2005) The Amber biomolecular simulation programs, *J Comput Chem 26*, 1668-1688.
38. Christen M, Hunenberger PH, Bakowies D, Baron R, Burgi R, Geerke DP, Heinz TN, Kastenholz MA, Krautler V, Oostenbrink C, Peter C, Trzesniak D, and van Gunsteren WF (2005) The GROMOS software for biomolecular simulation: GROMOS05, *J Comput Chem 26*, 1719-1751.
39. Forster F, Lasker K, Beck F, Nickell S, Sali A, and Baumeister W (2009) An Atomic Model AAA-ATPase/20S core particle sub-complex of the 26S proteasome, *Biochem Biophys Res Commun 388*, 228-233.
40. Nickell S, Beck F, Scheres SHW, Korinek A, Forster F, Lasker K, Mihalache O, Sun N, Nagy I, Sali A, Plitzko J, Carazo J, Mann M, and Baumeister W (2009) Insights into the Molecular Architecture of the 26S Proteasome, *Proc Natl Acad Sci U S A 29*, 11943-11947.
41. Lasker K, Sali A, and Wolfson HJ. Determining macromolecular assembly structures by molecular docking and fitting into an electron density map, *in press*.

42. Lasker K, Topf M, Sali A, and Wolfson H (2009) Inferential optimization for simultaneous fitting of multiple components into a cryoEM map of their assembly, *J Mol Biol 388*, 180-194.
43. Pettersen EF, Goddard TD, Huang CC, Couch GS, Greenblatt DM, Meng EC, and Ferrin TE (2004) UCSF Chimera--a visualization system for exploratory research and analysis, *J Comput Chem 25*, 1605-1612.
44. Kampmann M, and Blobel G (2009) Three-dimensional structure and flexibility of a membrane-coating module of the nuclear pore complex, *Nat Struct Mol Biol 16*, 782-788.
45. Sampathkumar P, Gheyi T, Miller SA, Bain K, Dickey M, Bonanno J, Kim S, Phillips J, Pieper U, Fernandez-Martinez J, Franke JD, Martel A, Tsuruta H, Atwell S, Thompson D, Emtage JS, Wasserman S, Rout MP, Sali A, Sauder JM, and Burley SK (Submitted) Structure of the C-terminal domain of Saccharomyces cerevisiae Nup133, a component of the Nuclear Pore Complex.
46. Putnam CD, Hammel M, Hura GL, and Tainer JA (2007) X-ray solution scattering (SAXS) combined with crystallography and computation: defining accurate macromolecular structures, conformations and assemblies in solution, *Q Rev Biophys 40*, 191-285.
47. Petoukhov MV, and Svergun DI (2007) Analysis of X-ray and neutron scattering from biomacromolecular solutions, *Curr Opin Struct Biol 17*, 562-571.
48. Schneidman-Duhovny D, Hammel A, and Sali A. (2010) FoXS: A Web Server for Rapid Computation and Fitting of SAXS Profiles, *Nucleic Acids Res, epub ahead of print*.
49. Forster F, Webb B, Krukenberg KA, Tsuruta H, Agard DA, and Sali A (2008) Integration of small-angle X-ray scattering data into structural modeling of proteins and their assemblies, *J Mol Biol 382*, 1089-1106.
50. Berman HM, Battistuz T, Bhat TN, Bluhm WF, Bourne PE, Burkhardt K, Feng Z, Gilliland GL, Iype L, Jain S, Fagan P, Marvin J, Padilla D, Ravichandran V, Schneider B, Thanki N, Weissig H, Westbrook JD, and Zardecki C (2002) The Protein Data Bank, *Acta Crystallogr D Biol Crystallogr 58*, 899-907.
51. Robinson RC, Turbedsky K, Kaiser DA, Marchand JB, Higgs HN, Choe S, and Pollard TD (2001) Crystal structure of Arp2/3 complex, *Science 294*, 1679-1684.
52. DeLano WL (2002) The PyMOL molecular graphics system, *Version 1.2r3pre, Schrödinger, LLC.*
53. Galassi M, Davies J, Theiler J, Gough B, Jungman G, Booth M, and Rossi F (2002) *GNU Scientific Library*.
54. Topf M, Lasker K, Webb B, Wolfson H, Chiu W, and Sali A (2008) Protein structure fitting and refinement guided by cryo-EM density, *Structure 16*, 295-307.

Tables

Table 1: IMP Modules

Module name	Description
Basic modules	
`core`	Basic functionality commonly used in structural modeling, including representation of particles as rigid bodies, commonly used restraints such as distance, excluded volume and connectivity**(*16*)** and frequently used optimizers such as Monte Carlo and conjugate gradients.
`algebra`	General-purpose algebraic and geometric methods, including principal component analysis of attributes, geometric alignment between two sets of 3D coordinates and geometric manipulations of spheres, cones, cylinders and cubes.
`display`	Tools for displaying and exporting of IMP data, such as intermediate models in an optimization process, in PDB**(*50*)**, Chimera**(*43*)** or PyMol**(*52*)** format.
`statistics`	Basic statistics tools, including *k*-means clustering, Gaussian mixture model clustering and histogram calculation.
`gsl`	Interfaces to allow algorithms from the GNU Scientific Library**(*53*)**, including simplex, quasi-Newton and conjugate gradients optimizers, to be used in IMP.
`container`	Tools and algorithms for manipulating subsets of the system's particles, such as maintaining a list of all pairs of particles that are spatially close.
Structural modeling modules	
`atom`	Tools for manipulating atoms and proteins. The module allows molecules to be read or written in PDB format**(*50*)** and scored using force fields such as CHARMM**(*36*)**. It also provides molecular dynamics and Brownian dynamics optimizers.
`em`	Integration of 2D and 3D EM data into the integrative modeling procedure. The module provides functionality to read and write EM density maps in MRC, X-PLOR, Spider and EM formats, to simulate density maps from a set of particles, and to represent the EM quality-of-fit as a restraint**(*54*)**.
`modeller`	Interface to the MODELLER**(*31, 32*)** comparative modeling program. The module allows for MODELLER models and restraints to be imported into IMP, and for IMP restraints to be used with the MODELLER optimizers, or *vice versa*.
`saxs`	Integration of SAXS data into the integrative modeling procedure. The module reads and writes SAXS profiles, and provides a restraint that scores a set of particles on their fit to an experimental profile**(*48, 49*)**.
`restrainer`	High-level interface for setting up an integrative modeling procedure, reading the representation of the system and the sources of input information from a pair of XML files.
`domino`	Implementation of an inferential message-passing optimization procedure**(*42*)**. The module provides functionality to build a graphical model of the defined scoring function, and to decompose the graph into a tree on which a message-passing sampling procedure is performed.
`multifit`	Tools for fitting multiple proteins into their assembly density map. The main functionality includes: (i) fitting a single protein into its density based on point-alignment, principal component matching or fast-Fourier transform search, (ii) combinatorial consideration of fitting solutions of multiple components for generating an assembly model, and (iii) modeling of cyclic symmetric assemblies**(*41*)**.
Support modules	
`benchmark`	A set of benchmarks of the IMP software, to ensure that the algorithms perform optimally.
`example`	Examples for developers on how to implement new IMP functionality.
`helper`	High-level functionality to assist in setting up and manipulating a system, including simplified interfaces for creating restraints such as EM, connectivity and excluded volume.
`misc`	Miscellaneous and experimental functionality that has not been fully tested.
`test`	Procedures to help in testing the IMP software itself.

Figures

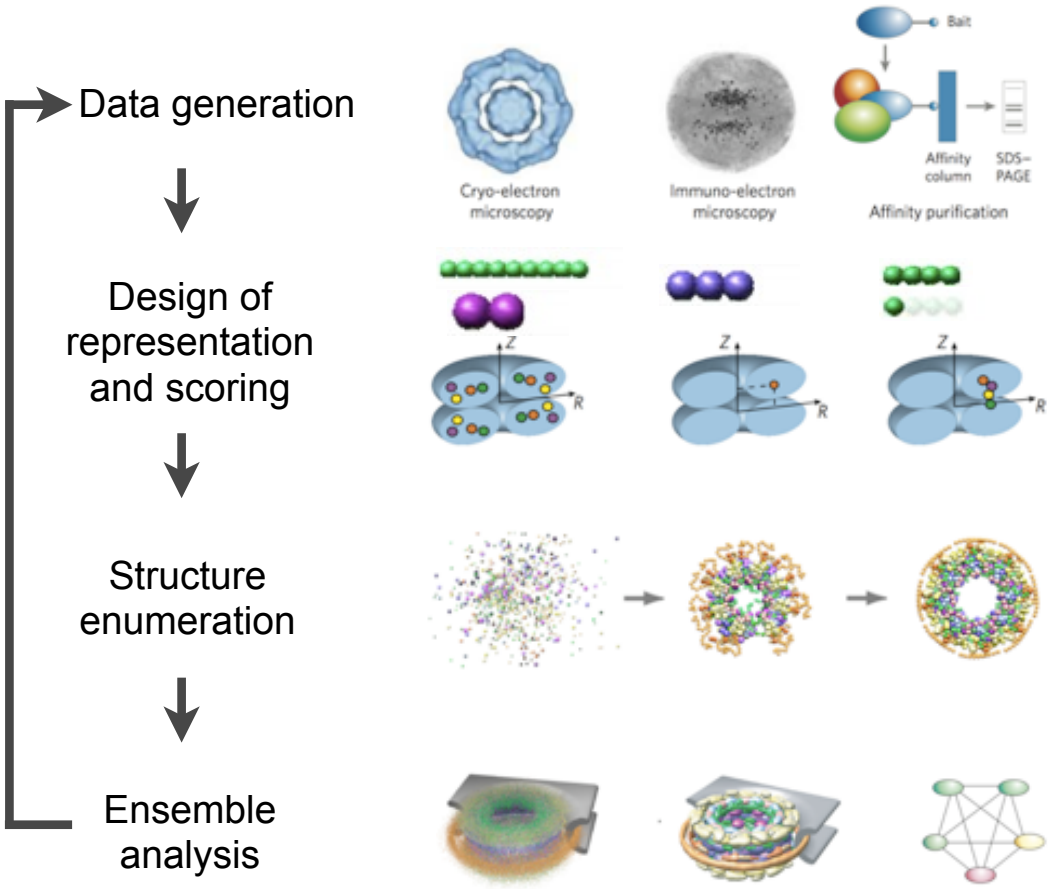

Figure 1. Integrative modeling protocol. After the datasets to be used are enumerated, a suitable representation is chosen for the system, and the input information is converted into a set of spatial restraints. Models are generated that are optimally consistent with the input information by optimizing a function of these restraints. Analysis of the resulting models informs about the model and data accuracy and may help guide further experiments. The protocol is demonstrated with the construction of a bead model of the NPC(*12*).

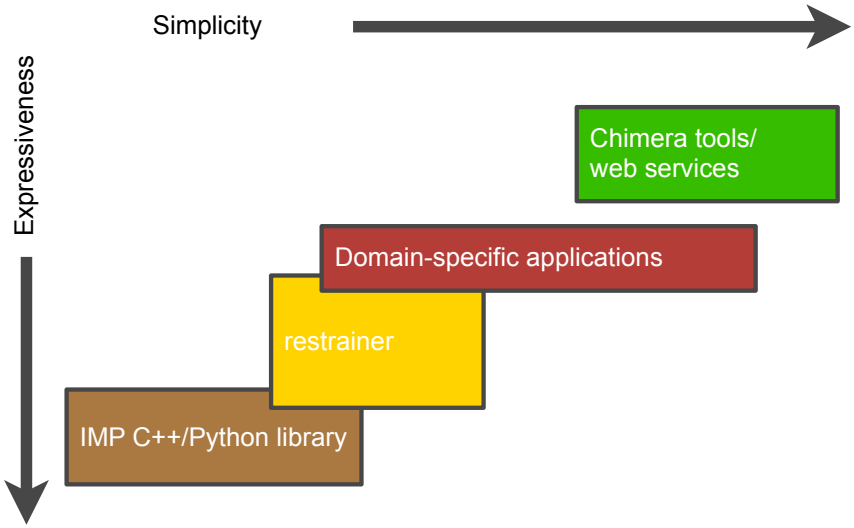

Figure 2. Overview of the IMP software. Components are displayed by simplicity (or user-friendliness) and expressiveness (or power). The core C++/Python library allows protocols to be designed from scratch; higher-level modules and applications provide more user-friendly interfaces.

```
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

# 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)

# 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
```

Figure 3. simple.py, a simple Python script that uses IMP to build a model consisting of two particles satisfying a harmonic distance restraint.

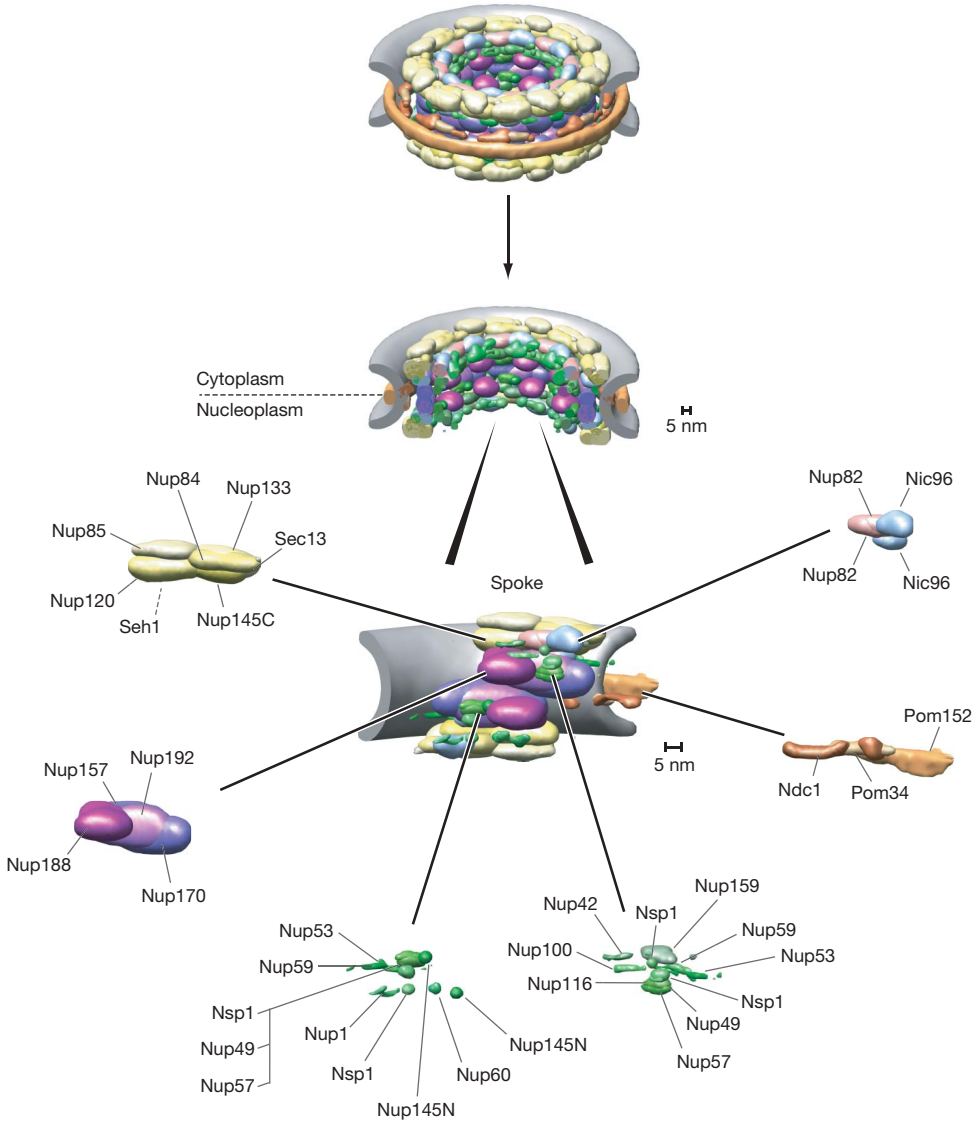

Figure 4. Division of the yeast NPC into subcomplexes(*13*); one such subcomplex is the Nup84 complex of seven proteins.

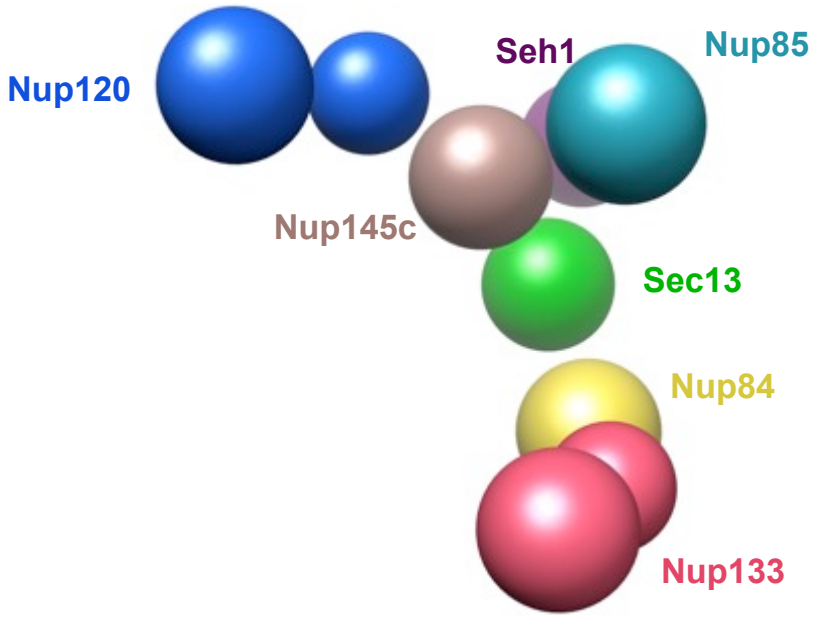

Figure 5. Bead model of the Nup84 complex generated by `restrainer`, based on yeast two-hybrid system data and excluded volume considerations.

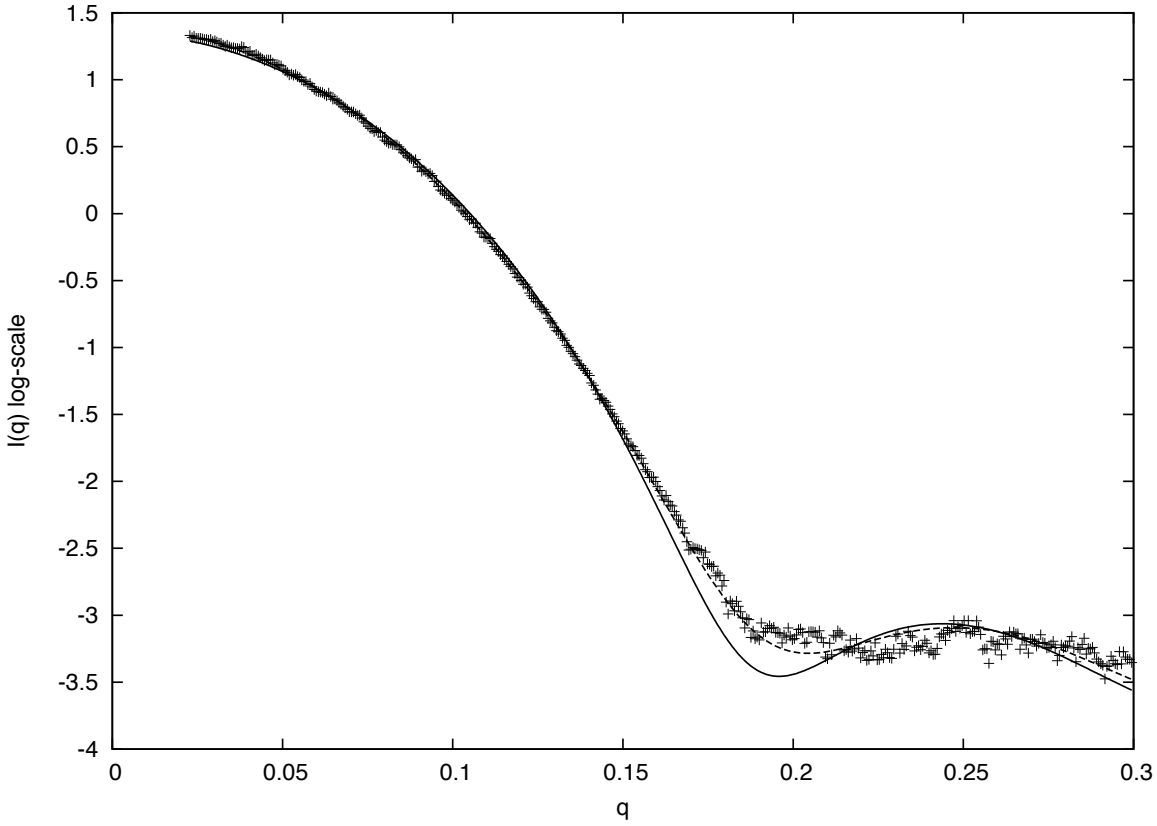

Figure 6. Fit of the 3kfo PDB structure (solid line) against the experimentally-determined SAXS profile of the same protein (points), using the FoXS web service at http://salilab.org/foxs(*48*). A plot of a comparative model's profile is also shown (dashed line).

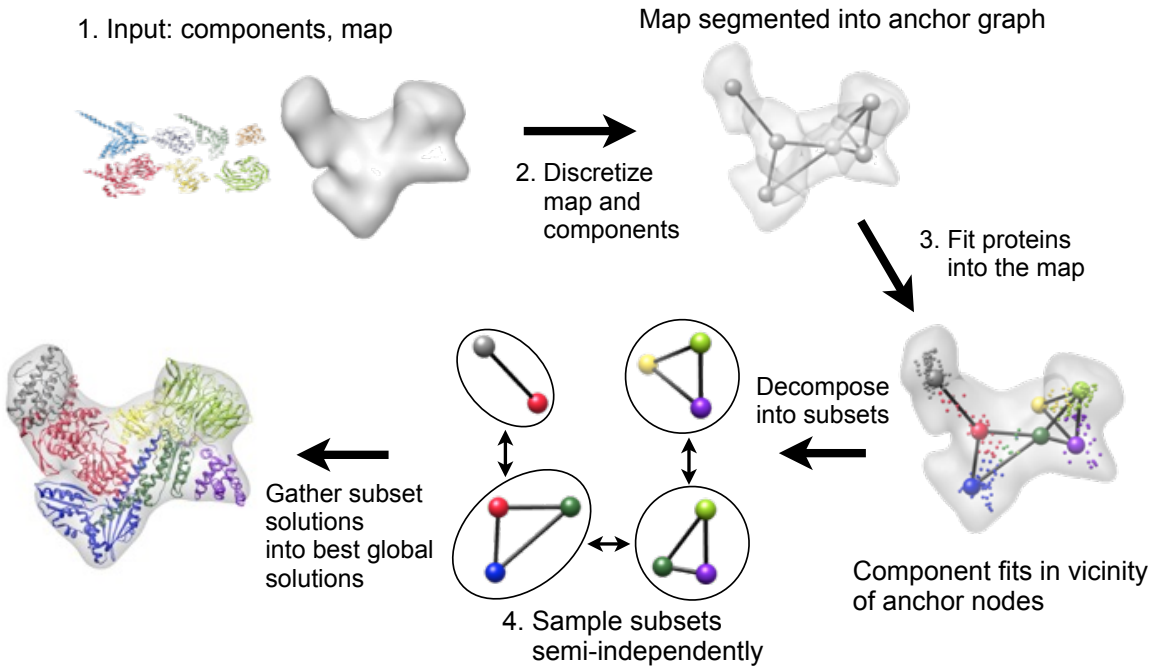

Figure 7. The `MultiFit` protocol(*42*). Protein subunits are fitted into a density map of the assembly by discretizing both the map and the components, locally fitting each protein, and efficiently combining the local fits into global solutions.