JDSHOT Documentation

Jeff Martin

April 13, 2009

1 Introduction

JDSHOT stands for Java D_n Symmetric Homo Oligomer Toolkit. This project contains sources for several tools useful for structure determination of symmetric proteins. One of the tools is essentially an implementation of the SYMBRANE algorithm adapted to work on D_n proteins. The sources here are specific to structure determination. Other more general code (i.e. reading/writing PDB files) is housed in a separate project called *share*.

This document describes the internal organization of code in the JDSHOT project and how to invoke its tools. Information regarding resource files and testing can be found in the documention for the *share* project where it is handled similarly.

2 Tools

Each tool is an executable command-line program housed in a JAR interface. These JARs reside in the *dist* directory of the project. To invoke a tool from the command-line, use a command of the following form:

```
java -jar path/to/jdshot-$(TOOL).jar $(ARGUMENTS)
```

where `$(TOOL)` is the name of a tool in the toolkit (each explained below) and `$(ARGUMENTS)` are the input supplied to the tool. All the tools use a unified command-line parsing module which offers a few useful features. Each tool is self-documenting. Simply invoke the tool without any arguments and it will return a list of the required and optional arguments along with a short description of each. The `--help` argument will also report the documentation. Additionally, these tools allow passing of arguments by a file instead of the command line. Use the `--file $PATH` argument to supply a file from which to read arguments. Argument files are the preferred way to invoke these tools simply because they are easy to edit and they can be saved in SVN. An example argument file is provided for each tool.

It is also possible to invoke these tools during development from the *bin* folder. Simply set the classpath to the bin folder and invoke the appropriate tool by its class:

```
java -cp path/to/bin edu.duke.donaldLab.jdshot.$(TOOL)Main $(ARGUMENTS)
```

All of the tools in the toolkit are described briefly below.

Search	SYMBRANE implementation. Multi-threaded.
Analyze	Calculates NOE satisfaction scores and other statistics for the set of grid cells output by an iteration of the search tool. Multi-threaded.
StericFilter	Filters a set of cells/points such that representative structures with significant backbone clashes are removed. The significance level is a configurable parameter. Multi-threaded.
Cluster	Performs clustering on a set of grid cells/points. Single-threaded.
Minimize	Invokes CNS to minimize representative structures for a set of grid cells/points. Single-threaded or MPI-cluster-runnable.

Test Runs the suite of unit tests for the project.

Compute This is sort of a catch-all tool to perform small tasks that don't need a separate tool. It accepts a few "standard" arguments such as paths to PDB files and NOEs, but it really doesn't have a stable interface. The behavior of this tool is configured primarily by a programmer at compile-time. It serves as a quick prototyping environment similar to scripting environments. The source for this tool is also very messy as it changes rapidly to perform different functions.

RigidBody Leftover code from a class project in computational geometry. This isn't really part of the toolkit, but it is listed here for completeness. It's safe to ignore this code. It will be eventually removed.

3 Inputs

While each tool has a slightly different set of input parameters, some inputs are common to many tools. These are listed below:

monomerPath Path to a PDB file containing single subunit

oligomerPath Path to a PDB file containing the full number of subunits arranged in their correct configuration

noesPath Path to a MR file containing assigned NOEs. Comments are also allowed.

subunitOrder This one is a little less straightfoward and will require some explanation. To make the implementation simpler, much of the code that deals with oligomer structures assumes that the subunits are provided in ring-order. Meaning, choose an arbitrary subunit to be subunit `A`. Then the adjacent subunit clockwise to `A` becomes `B` and so on around the ring. This argument provides a mapping between the names used in the PDB file, and this ring-order that the tools expect. It is represented as a character string of length equal to the number of subunits in the protein. The first character of the string gives the name of the subunit in the PDB file to be labeled `A`. The second character gives the name of the subunit in the PDB file to be labeled `B` and so on. For example, a subunit order might be `ABCD` if the protein is already labeled in ring-order. If not, the order input might be `ACBD`.

4 Packages

All packages are under the root package/namespace: edu.duke.donaldLab.jdshot

(root) The root package houses the main functions for each of tools in the toolkit

analyze code for the analysis tool

cluster code for the cluster tool

compute code for the compute tool

grid Contains classes for representing and dealing with grid cells/points. Much of this code is specific to D_n symmetry.

minimize code for the minimize tool

rigidBody code for the RigidBody tool

search code for the search tool

stericFilter code for the steric filter tool

test unit tests code. These classes are automatically excluded from the distribution JARs.

5 Compiling

Compiling the distribution files (the end-user JARs) is handled through Ant build scripting. The *dist* task performs the full compilation and creation of the distribution files (JARs).