ARGONNE NATIONAL LABORATORY
9700 South Cass Avenue
Argonne, IL 60439

ANL/MCS-TM-214

STALK Users Guide

by

*David Levine,[*] Michael Facello,[†] Philip Hallstrom,[‡]
Greg Reeder,[‡] Brian Walenz,[‡] and Fred Stevens[§]*

Mathematics and Computer Science Division

Technical Memorandum No. 214

July 1996

This work was supported by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Computational and Technology Research, U.S. Department of Energy, under Contract W-31-109-Eng-38, and by the Office of Health and Environmental Research, U.S. Department of Energy, under Contract W-31-109-Eng-38.

[*]Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Illinois 60439.

[†]Computer Science Department, University of Illinois, Champaign, Illinois.

[‡]Science and Engineering Research Semester Program, Argonne National Laboratory, Argonne, Illinois 60439.

[§]Center for Mechanistic Biology and Biotechnology, Argonne National Laboratory, Argonne, Illinois 60439.

Contents

1 Introduction **1**

2 Running a Job **1**
2.1 Running without Visualization 1
2.2 Running with Visualization 2

3 `stalk.ga`: The Docking Program **3**
3.1 Energy Computation 3
3.1.1 Basic Energy Function 3
3.1.2 Partitioning 4
3.1.3 Overlap 6
3.1.4 Intramolecular Energy 6
3.1.5 Future Energy Function Work 6
3.2 Rotamer Initialization 7
3.3 Genetic Algorithm 7

4 `stalk.cave`: The Graphics Program **7**
4.1 Wand Controls 9
4.2 The Main Menu 9
4.3 The Sidechain Menu 10
4.4 The Restart Options Menu 10

5 Files **11**
5.1 Input Files 11
5.1.1 Data Files 11
5.1.2 Parameter File 11
5.1.3 Rotamer File 13
5.2 Output Files 13
5.2.1 Solution Files 14
5.2.2 Standard Output 14
5.2.3 Standard Error 14

Appendix A: Sample .car File **15**

Appendix B: Sample Rotamer File **17**

Appendix C: Sample Parameter File **19**

Appendix D: Expected Sidechain Structures **21**

Appendix E: Miscellaneous Notes **23**

References 24

STALK Users Guide

by

David Levine, Michael Facello, Philip Hallstrom,
Greg Reeder, Brian Walenz, and Fred Stevens

1 Introduction

STALK is a system that models molecular docking between two proteins. A problem is posed as an optimization problem where the objective is to minimize the free energy of the molecular system by maximizing the intermolecular interaction energy between the two molecules. The possible number of conformations between the two molecules can be very large. A parallel genetic algorithm (GA) is used to explore the conformation space and identify the low-energy molecular configurations. The CAVE, a virtual reality environment, can be used to visualize and interact with the system while it is executing.

STALK consists of two programs: `stalk.ga` the docking program that runs the GA, and `stalk.cave`, the visualization program. The visualization component is optional.

2 Running a Job

In this document we assume you have already compiled and linked `stalk.ga` (and `stalk.cave` if you intend to use the visualization program). Details on the compilation process and file directory structure are given in the STALK Programmers Guide [LeFaHaReWaSt95]. Currently `stalk.ga` can be executed on an IBM RS/6000 (including the IBM SP parallel computer), Sun SparcStation, and SGI workstation. The program is written in standard Fortran and should run on most machines if a version of MPI and the PGAPack GA library are available. See [LeFaHaReWaSt95] for more details. Currently, `stalk.cave` runs *only* on SGI workstations.

2.1 Running without Visualization

The docking program `stalk.ga` is run via the Perl script `bin/rstalk.pl`. This has the advantage that the script sets a number of default values and handles any machine-specific setup needed.

When running on the IBM SP (where the path begins `/sphome/STALK`) the script `bin/sp_stalk.pl` is used to schedule/run a job. If the job is interactive, `sp_stalk.pl` is used to schedule the job and `rstalk.pl` is used after the job has been scheduled to run the job. If the job is batch, `sp_stalk.pl` both schedules *and* runs the job.

Below we give several examples of machine-specific usage. A full list of parameters to `bin/rstalk.pl` and `bin/sp_stalk.pl` is available by calling either with no parameters.

- Interactive execution on the IBM SP

```
cd /sphome/STALK
bin/sp_stalk -numnodes 8 -timelimit 60 -cac molecdoc -interactive
```

```
rlogin spnodeXXX
bin/rstalk -gaarch rs6000 -device mpl -novis -if input/ribo.in \
       -interactive
```

- Batch execution on the IBM SP

```
cd /sphome/STALK
bin/sp_stalk -numnodes 8 -timelimit 60 -cac molecdoc -gaarch rs6000 \
         -device mpl -novis -if input/ribo.in -of riboout -batch
```

- Interactive execution on a Sun workstation network

```
cd /sphome/STALK
bin/rstalk -gaarch sun4 -device p4 -novis -if input/stalk.in \
         -of riboout -interactive
```

2.2 Running with Visualization

This section explains how to run the GA algorithm on the IBM SP with the CAVE visualization component. To maintain GA efficiency, two different communication protocols are used, MPI on the SP and p4 between the SP and the CAVE. This is handled automatically by the Nexus implementation of MPI.

Startup consists of starting `stalk.ga` on the IBM SP, and starting `stalk.cave` on the CAVE machine. Nexus starts `stalk.ga` in a manner similar to the way that a normal MPI job is started. The user defines a set of environment variables that tell how many processes to start, and what nodes to start them on.

CAVE startup is handled by telling Nexus to additionally startup the CAVE node, in this case, `alaska.mcs.anl.gov`. Nexus determines what to start on the CAVE node by referring to the Nexus database file (see below).

Figure 1 is a shell script that defines the environment and starts both `stalk.ga` and `stalk.cave`.

The magic here is in the command line ("`bin/stalk.ga.sp2.bis -mpi ...`"). This tells Nexus four things:

- `-mpi` tells Nexus that the remaining arguments are for it
- The Nexus database file is `demo.rdb`
- In addition to the MPI job startup, start a job on alaska.mcs.anl.gov
- Do not use the name `ion.x` when starting the alaska.mcs.anl.gov job

The Nexus database file (Figure 2) tells Nexus what executable to run on various machines. The example shown has two entries, one for alaska.mcs.anl.gov which will start the CAVE, and one for flying.mcs.anl.gov which will start the ImmersaDesk. Which one is started depends on what is specified by the argument to `-nodes` in the command line.

```
#!/bin/sh

MP_HOSTFILE=/sphome/$LOGNAME/SPnodes.`getjid`
MP_PROCS=`cat $MP_HOSTFILE | wc -l`
MP_PULSE=0
MP_EUILIB=us

export MP_HOSTFILE MP_PROCS MP_PULSE MP_EUILIB

bin/stalk.ga.sp2.vis -mpi -dbfile demo.rdb -nodes alaska.mcs.anl.gov -nonameexpand
```

Figure 1: STALK Nexus startup script

```
alaska.mcs.anl.gov \
    startup_dir=/afs/fl/home/walenz/WORK/ION/bin \
    startup_exe=StartMolViewCave
flying.mcs.anl.gov \
    startup_dir=/afs/fl/home/walenz/WORK/ION/bin \
    startup_exe=StartMolViewIdesk
```

Figure 2: Sample Nexus database file

3 `stalk.ga`: The Docking Program

3.1 Energy Computation

The energy function is used to rank the GA strings according to which is the best solution to the problem at any iteration. The energy computation computes both the Coulombic and Van der Waals energy. In the absence of sidechain rotations, only the energy between the two molecules is computed, since the energy within a protein is constant. This is not true if sidechain rotations are allowed.

3.1.1 Basic Energy Function

The basic energy function computes the intermolecular energy. Let P_i be the set of atoms in protein i, with $|P_i| = n_i$. Let a_{ij} be the jth atom of P_i. Then, let

E_C^{inter} ... Coulombic energy

E_V^{inter} ... Van der Waals energy

q_{ij} ... charge of atom j in protein i

$d(a_{ij},\ a_{i'j'})$ ... Euclidean distance between the two atoms
D ... dielectric constant
A_{ij}, B_{ij} ... Van der Waals constants depending totally on the type of atom

$$
\begin{aligned}
E_C^{\text{inter}} &= \sum_{j=1}^{n_1} \sum_{j'=1}^{n_2} 0.322\, q_{1i}\, q_{2i'} \;/\; (D\, d(a_{1j},\ a_{2j'})) \\
E_V^{\text{inter}} &= \sum_{j=1}^{n_1} \sum_{j'=1}^{n_2} A_{1j}\, A_{2j'} \;/\; d(a_{1j},\ a_{2j'})^6 - B_{1j}\, B_{2j'} \;/\; d(a_{1j},\ a_{2j'})^{12} \\
E_{tot}^{\text{inter}} &= E_C^{\text{inter}} + E_V^{\text{inter}}
\end{aligned}
$$

This formula implies an $O(n_1 n_2)$ time algorithm for computing the energy, which is quite large. Section 3.1.2 presents an approximation technique that requires less computation.

3.1.2 Partitioning

To reduce the computation time, we use a three-dimensional subdivision of the space, with the size of a cell of the subdivision specified by a parameter. The cell containing an atom is computed for each atom. When the energy is being computed, only pairs of atoms that lie in the same cell or immediately adjacent cells contribute to the energy sum.

As an example, a two-dimensional partition of partition size p is shown in Figure 3. When performing an energy calculation, we consider only atoms in our nearest neighbor boxes (8 boxes in two dimensions, 26 boxes in three dimensions) plus our own box.

The energy function using this partitioning scheme is defined as follows. Given a parameter d_{min} (indicating that if two atoms are within distance d_{min} of each other, they should be included in this sum), then

C_{ij} ... the set of atoms not in protein i within distance d_{min} of atom a_{ij},

$$
\begin{aligned}
E_C^{\text{inter}} &= \sum_{j=1}^{n_1} \sum_{a_{2j'} \in C_{1j}} 0.322\, q_{1j}\, q_{2j'} \;/\; (D\, d(a_{1j},\ a_{2j'})), \text{and} \\
E_V^{\text{inter}} &= \sum_{i=1}^{n_1} \sum_{a_{2j'} \in C_{1j}} A_{1j}\, A_{2j'} \;/\; d(a_{1j},\ a_{2j'})^6 + B_{1j}\, B_{2j'} \;/\; d(a_{1j},\ a_{2j'})^{12}.
\end{aligned}
$$

Note that this is not completely correct. Basically, the space is partitioned by a grid, where the partitions along each of the coordinate axes are separated by distance d_{min}. If two atoms are either in the same cell or adjacent cells, they are included in this sum. Thus, the C_{ij} used in the program is a superset of the C_{ij} defined here.

Some tests were performed to compare the straight energy computation with the subdivision energy computation. In the latter, a cutoff of 10 angstroms was used (i.e., the cell size was $10 \times 10 \times 10$). This

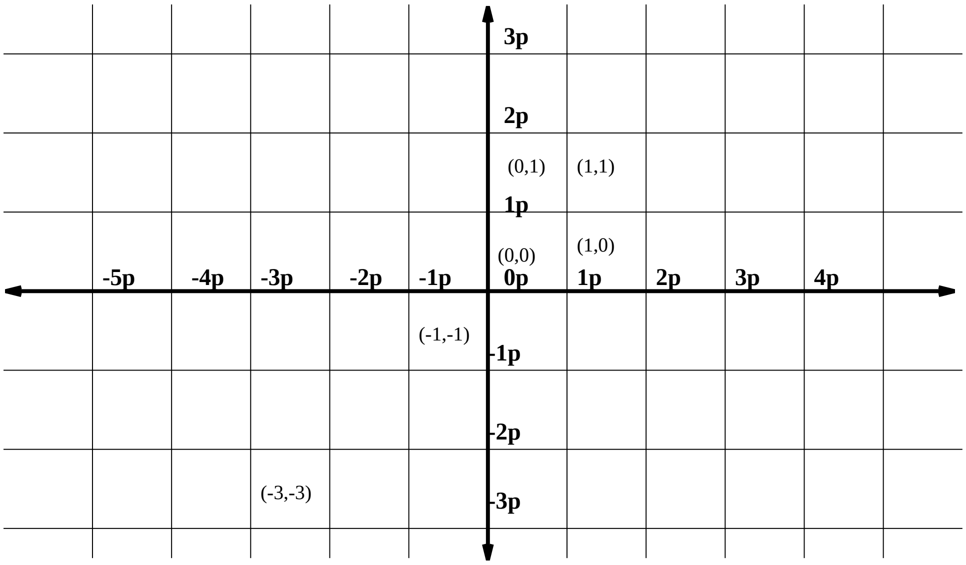

Figure 3: Energy function

implies that if two atoms are separated by at most 10 angstroms, they are guaranteed to be included in the computation. Some atom pairs with distance greater than 10 angstroms will also be included, but not all of them.

The energy was evaluated for chromosomes on a $20 \times 20 \times 20$ grid, where each of the translation elements of the chromosome ranged between 0 and 100 in increments of 5. No rotation was performed in these tests. The results of the tests showed that the latter computation used far fewer pairs of atoms to compute the energy, thus saving a significant amount of time, while still computing a similar energy function. Below are some statistics on the difference between the two computations.

- Minimum difference = -0.048323
- Maximum difference = 1.317550
- Average difference = 0.090510

Since the best score is -37.799883, these differences seem fairly insignificant relative to the full computation. Below are some statistics on the number of pairs evaluated.

- Total number of chromosomes tested = 25,492
- Total number of pairs = 453,560
- Averge number of pairs evaluated = 8,724
- Number of configurations with no atoms within 10 angstroms = 7,272
- Averge number of pairs not including configurations with 0 pairs = 11,111

3.1.3 Overlap

A conformation could cause the two proteins to overlap. The program handles this situation in one of two ways. The first is that for every pair of atoms that intersect, that is, the distance between their centers is less than a user-defined parameter, a penalty (also user-defined) is added to the energy. The idea of the penalty methods is that a conformation may overlap only a little and thus should not be penalized as much as a conformation that overlaps a lot. A good penalty amount still needs to be experimentally determined. The overlap penalty is handled in the function `pair_energy`. The second method uses the fact that because the Van der Waals component of the energy is very high for atoms close together, this acts as a natural penalty.

3.1.4 Intramolecular Energy

In the absence of sidechain rotations, the positions of atoms relative to other atoms in the same protein remain the same, and thus the intramolecular energy is constant. Therefore, only the intermolecular energy is computed for the GA energy function. In the presence of sidechain rotations, however, the intramolecular energy can change and is therefore very important. The formulas are as follows.

$$\begin{aligned}
C'_{ij} & \quad \ldots \quad \text{the set of atoms in protein } i \text{ within distance } d_{min} \text{ of atom } a_{ij} \\
E_C^{\text{intra}} &= \sum_{i=1}^{2}\sum_{j=1}^{n_i}\sum_{a_{ij'}\in C'_{ij}} 0.322\, q_{ij}\, q_{ij'} \;/\; (D\, d(a_{ij},\, a_{ij'})) \\
E_V^{\text{intra}} &= \sum_{i=1}^{2}\sum_{j=1}^{n_i}\sum_{a_{ij'}\in C'_{ij}} A_{ij}\, A_{ij'} \;/\; d(a_{ij},\, a_{ij'})^6 + B_{ij}\, B_{ij'} \;/\; d(a_{ij},\, a_{ij'})^{12} \\
E_{tot}^{\text{intra}} &= E_C^{\text{intra}} + E_V^{\text{intra}} \\
E_{tot} &= E_{tot}^{\text{inter}} + E_{tot}^{\text{intra}}
\end{aligned}$$

The current method for computing this energy, however, is not correct, because bonded pairs of atoms are included in the computation. This apparently causes the Van der Waals energy to increase dramatically. Thus, the issue of bonded atoms needs to be handled. A function called `exclude_pair` has been included in `energy.F` that will indicate whether a pair of atoms should be excluded from the energy computation. This function can be used in the future. For now, there is a flag in the input parameter file that can turn off the intra-molecular energy computation.

3.1.5 Future Energy Function Work

In general, the basic energy computation is quite fast using the partitioning method. When sidechain rotations are added, the energy computation takes somewhat longer because of the increased number of transformation operations. If intramolecular energy computations are also included, the energy computation becomes comparatively long. One possible method for using these three levels of energy computation is to first use the basic energy computation, and run the GA for some number of generations, and then use the best results to seed another run of the GA using the energy function with sidechain rotations. The result of this run can then be used in a subsequent run using the full energy computation.

3.2 Rotamer Initialization

Up to four positions along a sidechain can be rotated. A *rotamer* is a set of up to four angle values that uniquely specify the position of the sidechain. See [PoRi87] for a full description. These angles are measured according to the standard convention for defining dihedral angles as given in [Ri81].

When rotamer initialization is performed, first a random rotamer is chosen for the particular type of sidechain being considered. Several rotamers can be specified in the input file, along with the probability that each will be chosen for initialization. After a rotamer is chosen, a rotation angle must to be computed for each of the rotation points that will transform the orientation given in the initial data set to that of the rotamer angles. The function `compute_chi_angle` computes the current angle at a bond in the sidechain. The difference between this and the rotamer angle is used to initialize the chromosome.

3.3 Genetic Algorithm

STALK uses the PGAPack [Le95a, Le95b] a general-purpose, data-structure-neutral, parallel genetic algorithm library.

Each string in the GA population represents a potential minimizer of the energy function. Each string has six parameters. The first three parameters represent the x-, y- and z-translations of the ligand. The second three parameters represent the $x-$, $y-$ and $z-$rotations about the center of mass of the protein. Each parameter is represented by using a floating-point number.

When a run is made, the center of mass of the protein is translated to the origin of the system and remains stationary throughout the run. The GA strings are generated randomly. The initial translation values are selected uniformly randomly from a parameterized box about the protein center of mass. The rotation values are also selected uniformly randomly; the $x-$ and $y-$rotations from the range $[-\pi,\ \pi]$, and the $z-$rotation from the range $[-\pi/2,\ \pi/2]$. From the six parameters and the initial coordinates of the ligand atoms, one can compute a new set of atom centers using linear transformations of the atom coordinates by using matrix operations.

4 `stalk.cave`: The Graphics Program

`stalk.cave` is a visualization program that can be used in conjunction with `stalk.ga`. It allows the user both to visualize the best conformation found by the genetic algorithm each iteration and to interact with the algorithm by specifying strings the user creates within a virtual reality environment for inclusion within the GA population.

The `stalk.cave` program currently runs only on SGI workstations and is used in conjunction with either the CAVE (see Figure 4) or CAVE simulator virtual reality environment.

The protein is positioned at a predefined location in the CAVE (currently the center of the front wall). The ligand is positioned at a location offset from this, determined by the STALK program and

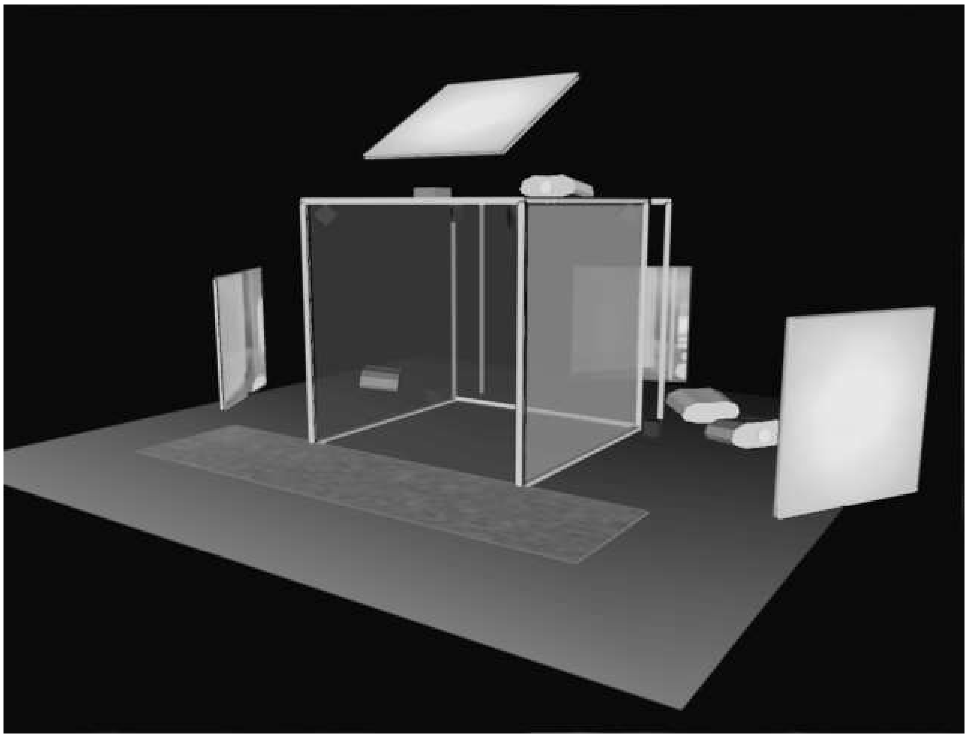

Figure 4: The CAVE virtual reality environment. Computer images sent to the projectors are folded by the mirrors and directed onto the CAVE walls and floor.

PGAPack. The current GA generation is displayed at the top center of the front wall, and information on the ligand's position and energy is displayed in the top right corner of the front wall.

The molecules are drawn as collections of spheres representing the atoms, connected by lines representing molecular bonds. The protein is initially drawn according to the following color code:

Element	*Color*
nitrogen	blue
carbon	green
hydrogen	white
oxygen	red
sulphur	yellow

The ligand is initially drawn with all molecules in magenta. All molecular bonds are drawn in cyan.

At each generation of the genetic algorithm, the ligand is drawn at the new position, and the generation and ligand information displays are updated.

4.1 Wand Controls

Button 1 of the wand allows the user to translate the image to a new location by pressing the button and moving the wand in the desired direction.

Button 2 allows the user to rotate the image by pressing the button and rotating the wand in the desired direction.

Button 3 brings up a menu that allows the user to change the operation of the simulation. Items are selected by highlighting the item by using button 1 of the wand to cycle forward, and button 2 to cycle backwards through the menu, and pressing button 3 to choose the item.

4.2 The Main Menu

The main menu is displayed when button 3 is pressed with no menu currently displayed. The items in the main menu are:

GA Running/Suspended Allows the user to stop the genetic algorithm in order to reposition the ligand, or to change the structure of the protein or ligand, and to then restart the algorithm.

Energy Evaluation Updates the energy value displayed, with the value for the ligand's new position. Available only when the genetic algorithm is suspended.

Ligand Info Displayed/Hidden Toggles whether or not the ligand information described above is displayed.

Protein Surface Displayed/Hidden Toggles whether or not the protein molecule is displayed with a cyan surface.

Ligand Color: Magenta/Full Color Toggles whether the ligand is displayed with the color scheme described above, or with all molecules magenta.

Sidechains Displayed/Hidden Toggles whether or not sidechains are displayed.

Wand Moves Ligand / Wand Changes View Toggles whether the wand controls move the entire image (the default) or just the ligand. Moving just the ligand changes its relative position, so that option is available only when the genetic algorithm is suspended.

Sidechain Operations Brings up a submenu for individual sidechain manipulation (see below). This item is available only if the sidechains are displayed.

End Simulation Ends both the CAVE program and the STALK program.

Exit Menu Removes the menu and returns wand controls to normal.

4.3 The Sidechain Menu

When the sidechain menu comes up, the first sidechain of the ligand molecule is highlighted in orange. The items in the menu are listed below.

Protein/Ligand Sidechains Indicates the molecule whose sidechains are being manipulated. Pressing button 3 toggles to the other molecule.

Sidechain X **Selected / Selecting Sidechain** X Allows the user to select a particular sidechain. X is the number of the sidechain selected. Pressing button 3 changes from the static display "Sidechain X Selected" to the dynamic display "Selecting Sidechain X". Button 1 then cycles forward through the sidechains, which are highlighted, and button 2 cycles backwards. Button 3 returns to the static display, with the new sidechain selected, and to normal menu control.

Angle: X **Degrees / Rotating:** X **Degrees** Allows the user to rotate the selected sidechain. X is the angle, in degrees, from the initial position that the sidechain is rotated. Pressing button 3 changes from the static display "Angle: X Degrees" to the dynamic display "Rotating: X Degrees". Button 1 then rotates the sidechain counterclockwise, and button 2 rotates it clockwise. Button 3 returns to the static display and to normal menu control.

Sidechain Hidden/Displayed Toggles whether or not the selected sidechain is displayed.

Sidechain Highlighted/Not Highlighted Toggles whether or not the selected sidechain is highlighted in orange. This item is available only if the selected sidechain is set to be displayed.

(NOTE: The selected sidechain will be highlighted in orange as long as it is selected and the sidechain menu is up, regardless of which display mode is selected for it.)

Exit Menu Removes the menu and returns wand controls to normal.

4.4 The Restart Options Menu

This menu is brought up when the genetic algorithm is restarted from the main menu. It controls how the algorithm is to be restarted. Choosing any item restarts the algorithm and exits the menu.

Replace Worst GA Pop Member The worst chromosome will be replaced by the ligand's current position and orientation.

Reseed Entire GA Pop The population will be reseeded with the ligand molecule's current position and orientation.

Restart without Changing The algorithm will be restarted without changing any of the chromosomes.

Figure 5 shows an example of the CAVE display.

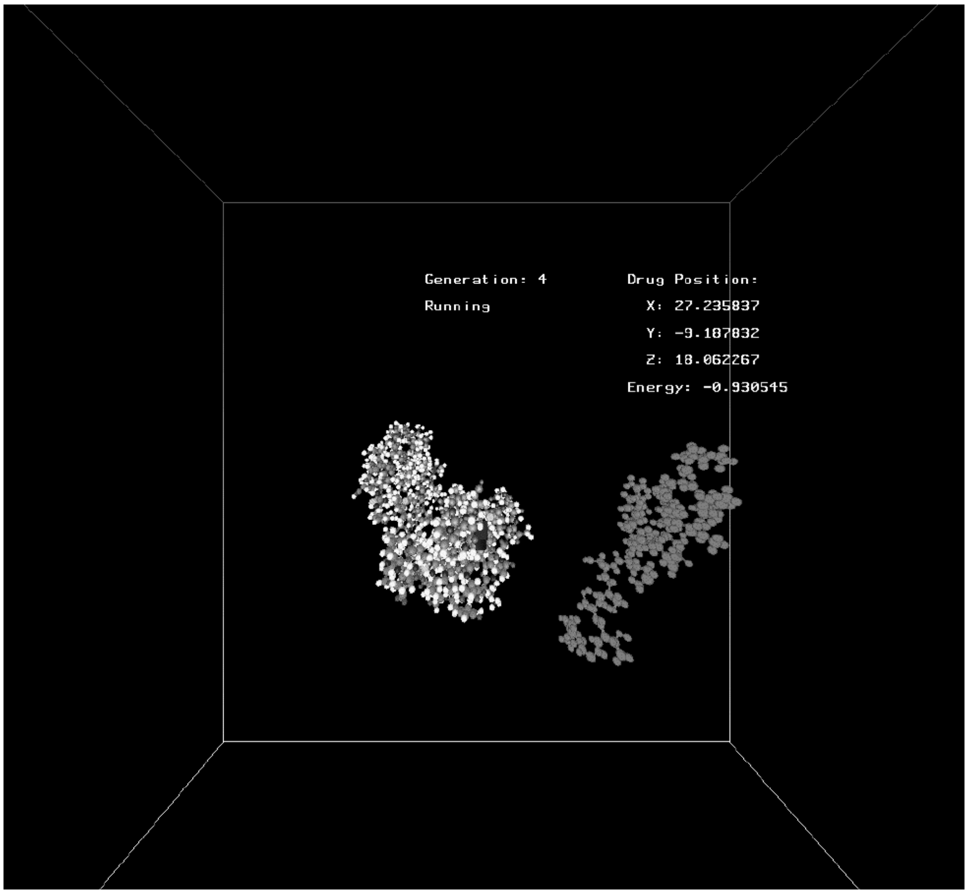

Figure 5: Example of the CAVE display

5 Files

5.1 Input Files

The inputs to STALK are (1) two .car files each of which contains the x, y, and z coordinates of all atoms in the protein, (2) a parameter file, and (3) an optional file of rotamer angles.

5.1.1 Data Files

The files describing the proteins are in Biosym's .car file format used in their *Discover* program. This format is a variant of the Brookhaven .pdb format. See Appendix A for an example.

5.1.2 Parameter File

Upon execution, the program will expect a set of parameters from the file `input/stalk.in` (see Appendix C). The format of this file is a string containing a description (for the convenience of the user) of the parameter followed by a line with the parameter value(s). All character values (including the comments) must have quotes around them. It is important that the correct data type be given. Here are the expected parameters.

- Number of objects (*integer*) – Currently, the program can model only two molecules.
- Molecule files (*character*) – Each file should be entered on a separate line. Currently the .car input format is supported. NOTE: The second molecule should be the smaller one because this will be the one that is transformed.
- Rotamer file (*character*) – If "Use rotamers" is true, use this rotamer file.
- Output format (*integer*) – If 0, do not create an output file. If 1, create a .pdb file. If 2, create a .car file.
- Echo energy (*logical*) – Display information about the energy evaluations.
- Partition size (*real*) – Size (in Å) used in subdividing the space into cells.
- Perform intramolecular energy calculation (*logical*) – Compute intramolecular energy. This parameter will increase the amount of time for each energy evaluation.
- Use penalty for overlap (*logical*) – Use a penalty value specified by the next parameter for each overlap encountered. Otherwise, set the energy to 999 and end the computation.
- Penalty amount (*real*) – If "Use penalty for overlap" is true, this amount will be added as a penalty to the Coulombic energy for every overlap encountered.
- Perform sidechain rotations (*logical*) – Allow sidechain rotations as variables to the GA.
- Use single angles (*logical*) – Use single or multiple angles for the sidechain rotation. Each sidechain has between 0 and 4 possible points of rotation. This variable will limit the choices to at most one each.
- Use rotamers (*logical*) – Enable rotamers.
- Temperature (*real*) – Not currently implemented.
- Coulomb_on (*real*) – The closest two atoms can be before they overlap.
- Dielectric (*real*) – Constant used in potential function.
- Maximum iterations (*integer*) – Number of generations the GA will run.
- Genetic algorithm population size (*integer*) – Population size.
- Generation Gap (*integer*) – Number of chromosomes in the current generation to replace when creating the next generation.

- Mutation probability (*real*) – Probability a mutation will occur. Mutation is checked only if crossover does not occur.
- Crossover probability (*real*) – Probability a crossover will occur.
- Lower and upper initialization bounds (*twelve reals*) – The lower and upper bound on ligand position and rotation initialization.
- Seed (*integer*) – Seed the random number generator. If the value is 0, a seed is generated based on the current system time. If nonzero, the value is used as the seed. The latter allows for repeatable runs.
- Eliminate duplicates (*logical*) – Eliminate duplicate strings each generation.
- Use restart operator (*logical*) – Specifies whether to restart the GA computation using a random variant of a subset of the strings in the current population.

5.1.3 Rotamer File

Rotamers are, optionally, used in the initialization phase of the GA. Each rotamer has a probability of being used for the initialization of a particular string. The rotamers may be destroyed through crossover or mutation operations of the GA.

The format for the file is a series of lines for each amino acid type,

$$
\begin{array}{llll}
\texttt{\#AAA}\ n_r & n_a & & \\
p_1 & a_{1,1} & \cdots & a_{1,n_a} \\
p_2 & a_{2,1} & \cdots & a_{2,n_a} \\
\vdots & & & \\
p_{n_r} & a_{n_r,1} & \cdots & a_{n_r,n_a},
\end{array}
$$

where `AAA` is the three-letter code of the amino acid, n_r is the number of rotamers for this type of amino acid, $0 \leq n_a \leq 4$ is the number of angles for this type of amino acid, and p_i is the probability that the ith rotamer will be chosen for the initialization of a chromosome. If $p' = 1.0 - \sum_{i=1} n_r p_i > 0.0$, there is a p' probability that the chromosome will be randomly initialized. Here $a_{i,j}$ is the jth angle of the ith rotamer. In general, $-\pi \leq a_{i,j} \leq \pi$. If $a_{i,j}$ is not in this range, that particular angle of the chromosome will be initialized randomly.

An example of this file is contained in Appendix B. Note that the amino acids must occur in the exact sequence given in this example. The three-letter codes merely serve as documentation for the user; for the program, the order is important.

5.2 Output Files

The primary output of STALK is a file containing the coordinates of the best conformation found. Additionally, both `stdout` and `stderr` may be written.

5.2.1 Solution Files

STALK can write the coordinates of the best conformation found in either .pdb or .car file format. The choice is specified in the parameter file (see Section 5.1.2).

5.2.2 Standard Output

STALK will occasionally write information about the progress of the GA to `stdout`. The amount and type are controlled by options in the parameter file (see Section 5.1.2).

5.2.3 Standard Error

Any errors will (most likely) be written to `stderr`, which should end up at the same place as `stdout`.

Appendix A: Sample .car File

The file format .car is used by Biosym's DISCOVER program. It is a variant of the Brookhaven .pdb format. The .car file gives each amino acid sequence in the protein. Specifically, for each amino acid it first gives the locations of the backbone atoms (the `N` to `O` atoms), followed by the atoms in the sidechain (usually beginning with `CB`, the beta carbon on the sidechain that attaches to the alpha carbon on the backbone), until the next `N` that starts the next amino acid sequence.

In a .car file, ten fields are given for each atom.

1. 1–5 the type of atom and level
2. 7–20 x coordinate of the atom (angstroms)
3. 22–35 y coordinate of the atom (angstroms)
4. 37–50 z coordinate of the atom (angstroms)
5. the residue type
6. the id of the residue type
7. the atom type
8. the atom type again
9. 75–80 partial charge on atom

The first character in the first column is the type of atom (e.g., `N` is nitrogen, `O` is oxygen,). The rest of the characters have to do with the level, where A, B, G, D, E, and Z correspond to α, β, γ, δ, ϵ, and ζ. These levels represent how many bond lengths away an atom is from a backbone atom.

If an atom is on the backbone, either we do not specify its level, or it is at the α level. If we give another atom (e.g., `HN`) it means that hydrogen is attached to nitrogen. If we give several atoms this way (e.g., `HN1`, `HN2`, `HN3`) it means they are all (in this example three) attached to the atom given by the second character.

The example below is from the file `data/kgk2.car`. It shows *all* of the atoms in the `LY+N` and `GLY` amino acids and *some* of the atoms in the `LY+C` amino acid.

```
'N' 10.251792948 -3.290935054 2.028567083 'LY+N' '1B' 'n4' 'N' -0.5000 1
'HN1' 9.739076078 -2.389121359 2.161643107 'LY+N' '1B' 'hn' 'H' 0.3600 2
'HN2' 10.552443744 -3.606320866 2.952220087 'LY+N' '1B' 'hn' 'H' 0.3600 3
'HN3' 11.166621453 -3.000052121 1.582450614 'LY+N' '1B' 'hn' 'H' 0.3600 4
'CA' 9.471758085 -4.284539111 1.227878688 'LY+N' '1B' 'ca' 'C' 0.3200 5
'HA' 8.990159669 -4.991126800 1.934372053 'LY+N' '1B' 'h' 'H' 0.1000 6
'C' 10.396905145 -5.183205299 0.349833379 'LY+N' '1B' 'c''' 'C' 0.3800 7
'O' 11.150974415 -5.967483090 0.925586956 'LY+N' '1B' 'o''' 'O' -0.3800 8
'CB' 8.362596107 -3.457812549 0.580144002 'LY+N' '1B' 'c2' 'C' -0.2000 9
```

```
'HB1' 7.814498884 -2.997844918 1.435134623 'LY+N' '1B' 'h' 'H' 0.1200 10
'HB2' 8.800071001 -2.593429416 0.031791125 'LY+N' '1B' 'h' 'H' 0.1200 11
'CG' 7.278980391 -4.106858742 -0.328703811 'LY+N' '1B' 'c2' 'C' -0.2000 12
'HG1' 7.692648815 -4.438057913 -1.300511198 'LY+N' '1B' 'h' 'H' 0.1200 13
'HG2' 6.825726796 -4.998022430 0.150244003 'LY+N' '1B' 'h' 'H' 0.1200 14
'CD' 6.222983723 -2.995712330 -0.518036302 'LY+N' '1B' 'c2' 'C' -0.2000 15
'HD1' 5.869596934 -2.759878057 0.508368600 'LY+N' '1B' 'h' 'H' 0.1200 16
'HD2' 6.742767019 -2.076779336 -0.881050876 'LY+N' '1B' 'h' 'H' 0.1200 17
'CE' 5.005274773 -3.245814989 -1.386812924 'LY+N' '1B' 'c2' 'C' -0.0800 18
'HE1' 5.272238275 -3.415875096 -2.453483391 'LY+N' '1B' 'h' 'H' 0.1900 19
'HE2' 4.434960841 -4.145579582 -1.064433132 'LY+N' '1B' 'h' 'H' 0.1900 20
'NZ' 4.205613165 -1.997087151 -1.201520184 'LY+N' '1B' 'n4' 'N' -0.5000 21
'HZ1' 4.694459848 -1.138466921 -1.517488125 'LY+N' '1B' 'hn' 'H' 0.3600 22
'HZ2' 3.236323902 -1.945788167 -1.603610441 'LY+N' '1B' 'hn' 'H' 0.3600 23
'HZ3' 4.042236688 -1.799366632 -0.199065867 'LY+N' '1B' 'hn' 'H' 0.3600 24
'N' 10.367540066 -5.099330000 -0.993733848 'GLY' '2B' 'n' 'N' -0.5000 25
'CA' 11.525489687 -5.536378105 -1.836538736 'GLY' '2B' 'cg' 'C' 0.0200 26
'HN' 9.750832573 -4.360444458 -1.347546714 'GLY' '2B' 'hn' 'H' 0.2800 27
'HA1' 12.218536265 -6.219846414 -1.305936539 'GLY' '2B' 'h' 'H' 0.1000 28
'HA2' 11.153741772 -6.110242504 -2.704949075 'GLY' '2B' 'h' 'H' 0.1000 29
'C' 12.288350237 -4.304739322 -2.364606351 'GLY' '2B' 'c''' 'C' 0.3800 30
'O' 12.076829574 -3.871585475 -3.496175156 'GLY' '2B' 'o''' 'O' -0.3800 31
'N' 13.039193505 -3.685723315 -1.457468903 'LY+C' '3B' 'n' 'N' -0.5000 32
'CA' 13.051420365 -2.208383677 -1.228957560 'LY+C' '3B' 'ca' 'C' 0.1200 33
'HN' 13.170465559 -4.242578330 -0.606190948 'LY+C' '3B' 'hn' 'H' 0.2800 34
'HA' 13.896945490 -1.761770817 -1.784856676 'LY+C' '3B' 'h' 'H' 0.1000 35
'C' 13.403227449 -2.122831712 0.299201694 'LY+C' '3B' 'c-' 'C' 0.1400 36
'OXT' 14.429952181 -1.509260102 0.648108579 'LY+C' '3B' 'o-' 'O' -0.5700 37
'O' 12.629358902 -2.627846859 1.148909119 'LY+C' '3B' 'o-' 'O' -0.5700 38
'end'
```

Appendix B: Sample Rotamer File

```
C  ****NOTE****  All angles *must* be in radians!!
C
C  The above rotamers were obtained from the paper by Ponder and Richards.
C
C  This file stores rotamers for the stalk program.  The format for each
C  the file is the following:  A line #XXX <number>  indicates the beginning of
C  the rotamers for sidechain type XXX.  There are <number> rotamers for this
C  type of sidechain.  Note that the sidechains should come in a specific
C  order, and even sidechains with no rotamers should be included with a value
C  0 for the number of rotamers.  The next <number> lines specify the rotamers,
C  with the format:
C
C <prob> <a1> <a2> ...
C
C  where <prob> is the probability that this rotamer should be used, and
C  <ai> is the angle for CHI_i specified in radians (at most four angles).
C  If the probabilities do not add up to 1 (but less than 1), a random
C  assignment will be made the rest of the time.  NOTE:  angles must be
C  in the range [-PI, PI].  If not, this indicates an unspecified angle,
C  and a random angle will be generated.
C
C  Comments can be added to the *beginning* of the file, such as this section,
C  by putting a capital letter "C" at the beginning of the file.
C
C  NOTE: Here are the number of chiral angles for the various sidechains
C
C GLY, ALA, PRO - 0
C       SER, CYS, THR, VAL - 1
C       ILE, LEU, ASP, ASN, HIS, PHE, TYR, TRP - 2
C MET, GLU, GLN - 3
C LYS, ARG - 4
C

#GLY 0 0
#ALA 0 0
#PRO 0 0
#SER 3 1
0.480  1.129227
0.286 -1.216493
0.235 -3.073522
```

```
#CYS 3 1
0.606 -1.137954
0.245 -3.134609
0.138  1.108283
#THR 3 1
0.479  1.094321
0.450 -1.041961
0.047 -2.958331
#VAL 3 1
0.671  3.028144
0.262 -1.106538
0.054  1.209512
#ILE 5 2
0.452 -1.062905  2.944368
0.183 -1.040215 -1.118755
0.161  1.076867  2.858847
0.129 -2.907716  2.897244
0.032 -3.050833  1.258381
#LEU 4 2
0.639 -1.132718  3.071777
0.245 -3.078758  1.101302
0.048 -2.885027  2.935641
0.020  0.773180  1.054178
#ASP 6 2
0.303 -1.192059 -0.642281
0.213 -3.090975  0.022689
0.131 -1.172860  2.247982
0.115  1.115264 -0.118682
0.115 -3.052578 -2.736674
0.066  1.110028  0.938986
#ASN 3 2
0.477 -1.192059 -0.448549
0.336 -2.951349  0.068068
0.159  1.111774  0.041888
```

Appendix C: Sample Parameter File

```
'***** F I L E S ***** '
'# Number of proteins                                       (n_objects - integer)'
2
'# Input files ( 1. Protein 2. Ligand )                     (obj_files(*) - char)'
'/sphome/STALK/data/rnsr1.car'
'/sphome/STALK/data/rnss0.car'
'# Rotamer file                                                 (rot_file - char)'
'/sphome/STALK/data/some.rot'
'# Output format:  0) none 1) pdb 2) car                  (output_type - integer)'
0
'***** S T A L K   P A R A M E T E R S ***** '
'# Echo energy information                                (energy_echo - logical)'
.FALSE.
'# Partition size                                           (part_size - double)'
10.
'# Do intra energy calculation                        (do_intra_energy - logical)'
.FALSE.
'# Use penalty for overlap?                               (use_penalty - logical)'
.FALSE.
'# Penalty amount per overlap                             (penalty_amt - double)'
.02
'# Do sidechain rotations                               (do_sidechains - logical)'
.FALSE.
'# Use single angles for sidechain rotations             (single_angle - logical)'
.FALSE.
'# Use rotamers?                                         (use_rotamers - logical)'
.FALSE.
'# Temperature                                            (temperature - double)'
300.
'# Coulomb_on                                              (coulomb_on - double)'
1.5
'# Dielectric                                              (dielectric - double)'
1.0
'***** G A   P A R A M E T E R S ***** '
'# Maximum iterations                                        (num_gen - integer)'
4000
'# Population size                                           (n_chrom - integer)'
1000
'# Generation gap                                        (num_replace - integer)'
100
```

```
'# Mutation  probability                                    (mut_rate - double)'
0.001
'# Crossover probability                                  (cross_rate - double)'
0.9
'# Lower bound initialization                               (lower(6) - double)'
0.0 0.0 0.0 0.0 0.0 0.0
'# Upper bound initialization                               (upper(6) - double)'
3.0 3.0 3.0 3.1415 3.1415 3.1415
'# Random number seed                                        (i_seed - integer)'
0
'# Eliminate duplicates strings                            (elim_dup - logical)'
.TRUE.
'# Use restart operator                                     (restart - logical)'
.FALSE.
```

Appendix D: Expected Sidechain Structures

This section lists the expected ordering of atoms within each type of sidechain. The starred atoms are those used in chiral angle computations and sidechain rotations.

0 angles:

GLY: N HN CA HA1 HA2 C O

ALA: N HN CA HA C O CB HB1 HB2 HB3

PRO: N CA HA CD HD1 HD2 C O CB HB1 HB2 CG HG1 HG2

1 angles:

SER: N* HN CA* HA C O CB* HB1 HB2 OG* HG

CYS: N* HN CA* HA C O CB* HB1 HB2 SG*

THR: N* HN CA* HA C O CB* HB OG1* HG1 CG2 HG21 HG22 HG23
problem: OG1 or CG2?

VAL: N* HN CA* HA C O CB* HB CG1* HG11 HG12 HG13 CG2 HG21 HG22 HG23

2 angles:

ILE: N* HN CA* HA C O CB* HB CG1 HG11 HG12 CG2* HG21 HG22 HG23 CD1* HD11 HD12 HD13
problem: is CG1 or CG2 the one on the longer chain?

LEU: N* HN CA* HA C O CB* HB1 HB2 CG* HG CD1* HD11 HD12 HD13 CD2 HD21 HD22 HD23

ASP: N* HN CA* HA C O CB* HB1 HB2 CG* OD1* OD2

ASN: N* HN CA* HA C O CB* HB1 HB1 CG* OD1* ND2 HD21 HD22
problem: OD1 or ND2?

HIS: N* HN CA* HA C O CB* HB1 HB2 CG* ND1* CD2 HD2 CE1 HE1 NE2 HE2
problem: ND1 or CD2?

PHE: N* HN CA* HA C O CB* HB1 HB2 CG* CD1* HD1 CD2 HD2 CE1 HE1 CE2 HE2 CZ HZ

TYR: N* HN CA* HA C O CB* HB1 HB2 CG* CD1* HD1 CD2 HD2 CE1 HE1 CE2 HE2 CZ OH HH

TRP: N* HN CA* HA C O CB* HB1 HB2 CG* CD1* CD2 NE1 CE2 HD1 HE1 CE3 HE3 CZ2 HZ2 CZ3 HZ3 CH2 HH2
problem: CD1 or CD2?

3 angles:

MET: N* HN CA* HA C O CB* HB1 HB2 CG* HG1 HG2 SD* CE* HE1 HE2 HE3

GLU: N* HN CA* HA C O CB* HB1 HB2 CG* HG1 HG2 CD* OE1* OE2

GLN: N* HN CA* HA C O CB* HB1 HB2 CG* HG1 HG2 CD* OE1* NE2 HE21 HE22
problem: OE1 or NE2?

4 angles:

LYS: N* HN CA* HA C O CB* HB1 HB2 CG* HG1 HG2 CD* HD1 HD2 CE* HE1 HE2 NZ* HZ1 HZ2 HZ3

ARG: N* HN CA* HA C O CB* HB1 HB2 CG* HG1 HG2 CD* HD1 HD2 NE* HE CZ* NH1 HH11 HH12 NH2 HH21 HH22

Appendix E: Miscellaneous Notes

1. To measure χ_i, the ith chiral angle of sidechain t, compute the dihedral angle of `chi_angles`(t,i), `chi_angles`(t,$i+1$), `chi_angles`(t,$i+2$), and `chi_angles`(t,$i+3$).

2. In order to conform to standard Fortran, all character strings in the input file must be in quotes. The Perl program `util/convert.pl` will help quote files in .car format.

3. To perform sidechain rotations, a certain structure (i.e., number of atoms, etc.) is assumed for each type of sidechain. However, because of the way the pdb (or .car) files are computed, sometimes they do not agree with the standard textbook structure. There are many checks in STALK; and unless they have the correct structure, an error message is printed and they are excluded from the sidechain rotation procedure. Therefore, this error message means the structure does not agree with the standard definition. The program will still run, however, without rotating the ill-formed sidechains. For example:

```
ERROR:  Object 1 Residue 26  : CYS   10 atoms expected,  0 found.
ERROR:  Object 1 Residue 27  : ASN   14 atoms expected,  0 found.
```

4. `rnsr1.car` has 1,564 atoms, and `rnss0.car` has 290 atoms. The minimum energy for these proteins is -37.799883 and occurs at the following translation values (with 0 rotation):

```
[ -13.94523630  -4.53963773  11.23557447 0 0 0 ]
```

   This is for the positioning of the molecules after their centers of mass have been translated to the origin.

5. The line right before the first energy evaluation says

```
Orig:     -5.7510025939760    1.3387824585403   0.21165003205882
```

   This is the position of the ligand in the original .car file. If these numbers are the first three elements of the chromosome, with 0's in all the other locations, the energy is about -10. For other values closer to these values, the energy is not zero. It looks like all the chromosomes that occur during the run are too far away, and therefore, the energy is 0. If one increases the number of chromosomes or decreases the initialization size of the translation, one should get non-zero values.

   To test this using `dbx`, stop after the first loop in the function transform. This loop initializes the values of an array called chromosome. Use statements like

```
assign chromosome[1] = -5.7510025939760
```

   to set the values of this array, and then see what happens to the energy evaluation.

References

[Le95a] D. Levine. *Users Guide to the PGAPack Parallel Genetic Algorithm Library.* Argonne National Laboratory, ANL-95/18, January 1996.

[Le95b] D. Levine. A public-domain parallel genetic algorithm library. Available by anonymous `ftp` from `info.mcs.anl.gov` in the directory `pub/pgapack`, file `pgapack.tar.Z`

[LeFaHaReWaSt95] D. Levine, M. Facello, P. Hallstrom, G. Reeder, B. Walenz, and F. Stevens. *STALK Programmers Guide.* ANL/MCS-TM-215, Argonne National Laboratory, 1996.

[PoRi87] J. W. Ponder and F. M. Richards. Tertiary Templates for Proteins—Use of Packing Criteria in the Enumeration of Allowed Sequences for Different Structural Classes. *J. Mol. Biol.* **193** (1987), 775–791.

[Ri81] J S. Richardson. The Anatomy and Taxonomy of Protein Structure. *Advances in Protein Chemistry* **34** (1981).