Übung V: Protein Structure Comparison

17 and 24. January 2008

Please deal with the following three tasks and submit your answers to torda@zbh.uni-hamburg.de not later than 7. January 2008. You have longer than usual to do write a report. Do not be scared by the programming.
Please send your report as an ascii, word or pdf file. Please do not use staroffice.

1. In the lectures, a little algorithm to compare two structures was presented on slide [54]. Given some alignment of the residues this algorithm iteratively superimposes one structure onto the other and removes the worst aligned residue pair until the difference falls below a threshold. You can find an implementation in

   `/home/torda/uebung_comparison/src`

   a) Copy the source code into your home directory with a command like

   `cp -r /home/torda/uebung_comparison/src .`

   b) Change into that directory and check if you copied the following files:

   dpstrct.c
   dpstrct.h
   evalali.c
   evalali.h
   main.c
   *Makefil*e
   *superimpose.*c
   superimpose.h

   c) Now compile the source code with

   `make`

   This should work without any problems. If not, ask for help.
   You can use

   `make clean`

   to remove any executables, object files and backup files. This is sometimes necessary if you want to recompile.

Text and code originally written by Gundolf Schenk

d) The executable *evalali.x* takes three command line arguments: a threshold and two pdb files. The program will store a superimposed version of the first pdb file in the current directory. The output can be very long, so it is easiest to view it if you redirect it to a file (**`>./out`**) and look at the output with **`less ./out`**

The program is not fully functional yet, so open the files containing the source code (*.c and *.h) in your favourite editor. The code is spread across four modules (i.e. seven files). Begin by looking at the main module. It calls functions from the WURST package and from the module *evalali*, which in turn uses functions from the modules superimpose and dpstrct. Read the comments in the header files (*.h) for hints on what the modules do. To see how the functions are implemented, you may look at the C source (*.c).

The main routine loads a structure and a template from two pdb files. Their sequences are aligned globally using a substitution matrix (BLOSUM62) via the Needleman & Wunsch algorithm. The structure is then superimposed onto the template and "interesting" residues are highlighted in the sequence alignment by the function **`selectInterestingAtoms()`**.

Open the file *evalali.c* and look for the function **`selectInterestingAtoms()`**. The algorithm from the lectures is implemented here. First it creates the **`dp`** list from an alignment. The list is a C-array of **`struct dpStrct`**. See the file *dpstrct.h* for its definition. Then it superimposes the structure onto the template using the alignment information stored in **`dp`** and updates the distance information in **`dp`** and sorts it. Then some difference measure of the aligned alpha carbons is calculated and printed to stdout. In the following **`while`** loop, the **`dp`** list is shortened by the residue pair with the largest (C^{α}) distance; the superposition, the **`dp`** list and the difference measure are updated until the difference falls below a given threshold or less then four aligned residue pairs are left in the **`dp`** list. A minimum of three residue pairs are needed to calculate a rotation matrix.

e) Now, have a look at the function **`alphadiff()`**. As arguments it takes the structure, the template, the **`dp`** list and its length and a flag. This flag controls the kind of difference measure used on the alpha carbons. Your task is now to code up two difference measures, the root mean squared distance (RMSD) and the distance matrix error (DME). The DME is also known as root mean squared distance matrix difference. Remember from the lectures:

$$RMSD = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left|\vec{r}_i - \vec{r}_i'\right|^2}$$

$$DME = \sqrt{\frac{2}{N(N-1)}\sum_{i=1}^{N}\sum_{j=1}^{N}\left(\left|\vec{r}_i - \vec{r}_j\right| - \left|\vec{r}_i' - \vec{r}_j'\right|\right)^2}$$

Write your code for the two measures at the appropriate places inside the switch statement.

Replace the comments, the error messages and the exit call with your code. Some hints:
The alpha carbons of the whole protein are stored in an array of three-dimensional vectors called **`rp_ca`**, which is a member of the struct coord. Each vector has members x, y and z. Use the dp list to access the aligned residues via their position in rp_ca. See the function **`updateDpDist()`** in *dpstrct.c* for a sample usage. For definitions of the involved C-structs see files *dpstrct.h* and *coord.h*. For mathematical operations you may use anything you can find in the WURST package. Look in
/home/torda/uebung_comparison/wurst/src
or from the standard C library, e.g. *math.h* (see www.cppreference.com). If you would like a quick reminder of C syntax, look at
`www.num.math.uni-goettingen.de/Dokumentationen/C/Einfuehrung/`
There is also a fair interface description of the most relevant functions of the standard C library at the bottom of the right navigation frame.
Please obey some coding rules:

- write robust and fast code (sometimes a tradeoff)
- avoid code duplication
- comment your code sensibly
- use only ANSI-C (i.e. remove all compiler warnings and errors)
- indent in the format of the rest of the code (similar to Kernighan & Richie)

f) Compile two versions of the program,one using RMSD and one using DME as difference measure. To do so, find the lines where the **`alphadiff()`** function is called in the function **`selectInterestingAtoms()`** and make sure that their last parameters are YES. Then compile the program with **`make`**. Rename the executable evalali.x with **`mv evalali.x evalali_rmsd.x`**. Now, change the function's parameters to NO, recompile and rename with **`mv evalali.x evalali_dme.x`**. By now you should have two executables, which you might want to use with two structures, e.g. 1zik and 1et1.

g) Compare the algorithm from the lectures with the implementation in *evalali.c*. What is different (algorithmically) ? Do you think it is a serious difference ? What impact could it have on the result ?

2. Load the two proteins 1ECA and 1LHS from the PDB repository in UCSF Chimera.

a) Change to ribbon view. Superimpose the two molecules:
Tools>>Structure Comparison>>MatchMaker
The MatchMaker window should appear.

Select 1ECA as the reference structure and 1LHS as the structure to match. Use SmithWaterman as alignment algorithm and default values for the rest, and click 'OK'. The RMSD is given in the status bar of the main window.

Provided that the box 'show alignment(s) in MultAlign Viewer' was checked, you should find the sequence alignments in the popup MultAlign windows. You can save the alignment in FASTA format for your report:

From the MultiAlignViewer window,

File>>Save As...

Perform another fit for the molecules using the NeedlemanWunsch alignment algorithm. In your report, describe the change in RMSD, and explain why it is different.

b) Optimizing the superposition:

From the MultAlignViewer window,

Structure>>Match...

Again, select 1ECA as the reference structure, and 1LHS as the structure to match, and check the box 'Iterate by pruning long atom pairs'. Enter a number to the textfield of 'until no pair exceeds __ angstroms. Click 'Apply' to observe the change (Hints: You can start from 6 Å… , then scale it down to 4 Å… , 3 Å… , 2 Å… ...).

c) Assessing the fit:

From the MultAlignViewer window,

Structure>>Assess Match...

Select IECA as the reference structure and 1LHS as the structure to evaluate, click 'OK'. Select the Attrribute 'matchDist' and move one bar in the histogram to zero. Input a number (e.g. 2.0) for the second bar and make sure that the box 'between markers (inclusive)' is checked, click 'Apply'. Input another number (e.g. 1.0) for the second bar, and click 'Apply' again. You should see which residues fit better from the alignment. Include the alignment with selected residues highlighted in your report:

File>>Save EPS...

Switch to the 'Render' tab, select the attribute 'matchDist' again and set the red bar to 2.0, the white bar to 1.0, and the blue bar to zero. Click 'Apply'. Save an image of the coloured molecules.

d) Exploring the chemical features:

What are the differences between these two proteins (e.g. the ends, or certain parts between the helices)? Once the structures are superimposed, you can compare their chemical features more closely. Display the haem group of both proteins:

Select>>Structure>>ligand

Action>>Surface>>show

You should see the superimposed haem groups. Select the conserved residues of the molecules:

From the MultAlignViewer window,

Structure>>Select by Conservation...

Select the attributes of 'residues' and highlight both models. Pick 'mavPercentConvered' for the Select Attribute, and move the markers to the one end (100). Use default values for the rest, and click 'OK'.

Which residues are near the haem group, and how well are they conserved in sequence and geometry. Are the matched residues in each structure interacting with the Fe-porphyrin complex in the same way?

e) Other Structural Alignments:

Explore other matching criteria.

From the MultAlignViewer window,

Structure>>Match...

Try 'Match highly conserved residues only', which causes only the wellconserved (at least 80%) positions in the alignment to be used for the leastsquares fit, and different values for the 'Iterate by pruning long atom pairs until no pair exceeds [x] angstroms', which refers to an iterative fitting procedure. In each cycle, atom pairs are removed from the match list and the remaining pairs are fitted, until no matched pair is more than x angstroms apart. The atom pairs removed are either the 10% farthest apart of all pairs or the 50% farthest apart of all pairs exceeding the cutoff, whichever is the lesser number of pairs. The result is that the best matching "core" regions are maximally superimposed; conformationally dissimilar regions such as flexible loops are not included in the final fit, even though they may be aligned in the sequence alignment.

Make a note of how many residues are aligned, and the RMSD of the alignment.

f) In the MultAlignViewer window, you can quickly get an estimate of sequence similarity. Under 'Tools', you should find 'Percent identity...'. Note down the value for this pair of proteins. Looking at the sequence alignment, count the number of gaps.

The sequence similarity here is less than 25%. This is not very high, but the structures appear very similar. Would you expect this to be the case for all apirs of proteins? What other factor

determines the significance of an alignment? Although the sequence similarity is low, are there any clues as to why it works so well ?

3. Use your two programs from task 1 with 1ECA and 1LHS. Compare the results of the two programs to the result chimera produces when 'Iterate by pruning long atom pairs until no pair exceeds [x] angstroms' is checked in the 'Match Structures by Seq' dialog. Use your results from task 2 here. Describe any differences and why they might occur.