DECOMP: A PDB decomposition tool on the web

Rafael Ordog1,2, Zoltán Szabadka1,2,3, Vince Grolmusz1,2*  
1Protein Information Technology Group, Eötvös University, 1117 Budapest, Hungary; 2Uratim Ltd. 4400 Nyíregyháza, Hungary; 3Google Inc, Europe; Vince Grolmusz – Email: grolmusz@cs.elte.hu; * Corresponding author

Received February 22, 2009; accepted May 22, 2009; published July 27, 2009

Abstract:  
The protein databank (PDB) contains high quality structural data for computational structural biology investigations. We have earlier described a fast tool (the decom_pdb tool) for identifying and marking missing atoms and residues in PDB files. The tool also automatically decomposes PDB entries into separate files describing ligands and polypeptide chains. Here, we describe a web interface named DECOMP for the tool. Our program correctly identifies multi-monomer ligands, and the server also offers the preprocessed ligand-protein decomposition of the complete PDB for downloading (up to size: 5GB)

Availability: http://decomp.pitgroup.org

Keywords: PDB, web tool, decomposition, server, ligands, SEQRES

Background:  
The Protein Data Bank [1] started to function as the depository of the crystallographic data, complementing journal publications: researchers solved the structure of a protein, wrote a paper on the result, and deposited the data of the solution in the publicly available PDB. The irregularities of the structure deposited (such as lacking atomic coordinates, broken chains, unidentified substructures) are mostly remarked in the cited publications and also in the remark-fields of the PDB file. The textual annotations in the scientific publication elsewhere or in the remark-fields in the very same PDB-file, however, make the automatic processing of the protein-structures very difficult. This statement may be a little bit confusing, since atoms, carrying the HET label are not supposed to be in the peptide-chain, so those structures that contain HET atoms other than the oxygen of the water would qualify for being a complex. Unfortunately, this is not the case. Metal ions, modified residues (in a surprisingly large number), and small molecules added in the crystallization all contain heteroatoms, and they are frequently not considered to be ligands. With our decom_pdb program [2] protein-ligand complexes are identified reliably, and the ligands are deposited in separate files. Missing residues and atoms in chains are handled properly, that is, even if several atoms are missing from a chain our algorithm will still not recognize the parts as distinct chains. Placeholders are inserted into chains for missing residues/atoms (an example is given in Figure 2), denoting that the objects were not measured crystallographically, but according to the more reliable sequence information - they should be there. This way our algorithm "repairs" faulty PDB's, or recognizes that flexible chain sequences are present. We should remark, that missing atoms are usually a sign of mobile loop or string in the protein-crystal, since flexible atoms will not give usable electron density maps. Consequently, mapping missing atoms this way may help to automatically identify flexible protein parts. Ligands are identified without using the HET-atom labels, properly handling modified residues and small artifacts, due to crystallization protocols. CONECT records of the ligand-atoms are computed automatically (these records for the ligands generally are not present in the PDB file).

Figure 1: The DECOMP_PDB output-ligand 10gs.pdb.out.lig.3 contains the 3-monomer GLU-BCS-PG9 molecule correctly, in one single file, even if it contains three monomer ID’s.
Methodology:
Our program selects atoms from the PDB entry that are part of a protein or DNA chain. We do not use the chain-identifier for this purpose. However, we use SEQRES data and refined graph-theoretical algorithms described elsewhere [2]. It selects the water molecules, and removes them from the set of possible ligand atoms. Then metal and other small ions are selected, that will not be considered as ligands. A complete list of residue names that were considered as ions (so not as ligands) is given in the file ion_list.txt. All the remaining atoms will form the set of ligand atoms. Within this set, we use a graph-theory component detecting algorithm, so a ligand is defined as a connected component of the graph formed by the ligand atoms as vertices and the covalent bonds between the ligand atoms as the edges.

Functionality:
The DECOMP tool correctly identifies ligand molecules, even if they are composed of more than one monomers. For example, when decomposing PDB entry 10GS with options “Export ligands”, the file 10gs.pdb.out.lig.3 contains the 3-monomer GLU-BCS-PG9 molecule correctly (Figure 1).

Utility:
Provide a list of PDB codes in the appropriate box at the web server and check the desired options. The PDB codes should be separated either by “spaces” or “new line” characters. Press the “schedule job” button and the request will be inserted into a queue. Progress is monitored in the “Log window”. The result will be a link in the “Log window” to a tar.gz file. The result file contains one directory for each of the pdb’s listed. Each of these directories contains an error log with “.pdb.error” extension, the decomposed pdb file with “.pdb” extension, and if “Export ligands” or “Export ions” option was specified, than a separate file is present for each of the ligands or ions. An error file is presented if there was a fatal error while processing the PDB file. The result files are usually viewed by popular PDB viewer tools. A preprocessed, constantly updated compressed file can be downloaded with the results when the entire PDB file has been decomposed. The result files are stored for 3 days, and log files are stored for 30 days in the server.

Acknowledgements:
The authors acknowledge the partial funding of the following agencies: NKTH: OMFB-01295/2006 and OM-00219/2007 through Uratim Ltd. and OTKA NK-67867 through Eotvos University.

References:
[1] H M Berman et al., Nucl. Acids Res., (2000) 28: 235
[2] Z Szabadka, V Groflmusz, J. Mol. Graph. Mod. (2007) 25: 831

License statement: This is an open-access article, which permits unrestricted use, distribution, and reproduction in any medium, for non-commercial purposes, provided the original author and source are credited.

Edited by P. Kangueane