Fernandez-Recio

Assembly prediction in CASP14 with pyDock ab initio docking and scoring

M. Rosell\textsuperscript{1,2}, L.A. Rodriguez-Lumbraeras\textsuperscript{1,2} and J. Fernández-Recio\textsuperscript{1,2}

\textsuperscript{1} - Instituto de Ciencias de la Vid y del Vino (CSIC), \textsuperscript{2} - Barcelona Supercomputing Center (BSC)

juan.fernandezrecio@icv.es

Key: Auto:Y; CASP_serv:Y; Templ:Y; MSA:N; Fragm:N; Cont:N; Dist:N; Tors:N; DeepL:N; EMA:Y; MD:Y

In the past 3\textsuperscript{rd} common CASP-CAPRI Assembly Prediction challenge, our modeling approach, integrating \textit{ab initio} docking, template-based modeling, distance-based restraints, low-resolution structural data and symmetry constraints, yielded excellent performance, ranking 2\textsuperscript{nd} among CAPRI predictors, and 1\textsuperscript{st} among CAPRI scorers\textsuperscript{1}. Here we describe our participation in the CASP14 Assembly category, as part of the 4\textsuperscript{th} common CASP-CAPRI Assembly Prediction challenge (CAPRI Round 50). We have participated as human predictors, human scorers, and server scorers, in all the 18 proposed targets, consisting in four hetero-dimers (A1B1), six homo-dimers (A2), two homo-trimers (A3), two homo-tetramers (A4), one hetero-nonamer (A3B3C3), one homo-20mer (A20), one hetero-27mer (A6B3C12D6), and one homo-240mer (A240).

Methods

For each assembly, the models of the individual subunits were taken from the ZHANG, RaptorX, and QUARK CASP-hosted servers (only the best prediction for each server was used). In CAPRI target ID T170 (CASP target ID H1060), the experimental structures of two of the subunits were available (see more details in Results section). In two other cases (T165/H1036 and T177/H1081), there were not available models at the CASP-hosted servers for some of the subunits, so we modelled them with MODELLER\textsuperscript{v9.19}.

Using the available structural models of the individual subunits as above described, we modelled all or some binary interactions in the assembly by \textit{ab initio} docking (usually docking one pair of models from each CASP-host server). As human group we applied our pyDock\textsuperscript{2} docking and scoring pipeline, in which we used FTDock (electrostatics on; 0.7 Å grid resolution) and ZDOCK 2.1 to generate 10,000 and 2,000 rigid-body docking poses, respectively, which were merged in a single pool for subsequent pyDock energy-based scoring. We also participated with our pyDockWEB server\textsuperscript{3}. In homo-oligomers, docking poses not satisfying the expected symmetry (e.g. $C_2$ for homo-dimers, $C_3$ for homo-timers, etc.) were removed.

Additionally, we checked if there were available templates for all or part of the assembly interfaces. First, we used BLAST for this. In parallel, for each complex we searched for oligomeric templates from the top five released predictions from the ZHANG, QUARK, RaptorX, MULTICOM-CONSTRUCTand ROSETTA CASP-hosted servers. These monomeric models were superimposed onto the corresponding subunits of each selected template and minimized with AMBER 12.

Finally, all the generated models (either \textit{ab initio} or template-based) were scored with pyDock, and sorted according to the summation of the binding energy of all possible interfaces. The number of available templates and their reliability determined the percentage of template-
based complex models included in the top 5 and 10 submitted models. Finally, we eliminated the redundant predictions and minimized the top ten submitted models.

In the scorers experiment, we first removed models with more than 250 clashes (i.e., intermolecular pairs of atoms closer than 3 Å). Then, we applied pyDock scoring and used the same criteria to rank the docking models as in predictors (i.e. in case of reliable templates we favored models similar to such templates, we checked for symmetry, we applied ad-hoc distance restraints for specific targets, etc., more details in the Results section). As human scorers we introduced more human intervention than as server scorers, i.e., removing loops with non-realistic conformations, and re-scoring some of these models afterwards.

Results
We submitted models generated only by ab initio docking in those targets for which we could not find available templates (T169/T1054, T172/H1066, T173/H1069, T174/T1070, T178/T1083, T179/T1087, and T181/H1103). In the case of T174/T1070, all the template-based models we built had clashes, so as predictors only ab initio docking models were submitted, but as scorers we favored models similar to the available templates. In the case of T181/H1103, additional restraints were applied to remove poses clashing with the membrane regions.

But for the majority of targets, we could find potentially suitable templates for all or some of the predicted interfaces. In many cases, we generated models by ab initio docking and by template-based modeling independently, and the final proportion of models derived from these two approaches was determined by pyDock scoring and/or by the reliability of the available templates. Thus, ab initio docking was favored in targets T164/T1032 and T176/T1078, while template-based modeling was favored in targets T166/H1045, T167/T1050, and T168/T1052.

On the other side, in target T180/T1099, consisting in the assembly of a virus capsid with icosahedral symmetry, only template-based modeling was used.

In the remaining targets, in order to build the full assembly we combined template-based docking for some interfaces and ab initio docking for the other ones. This is the case of T177/H1081 (A20), in which the homo-decamer (A10) was modelled based on available templates followed by pyDock scoring, and the final assembly was built by docking two decamers. Similarly, in homo-tetrameric targets T171/T1063 and T175/T1073 (A4), one of the dimeric interfaces was modelled based on available templates and then the modelled dimers were docked to built the full assembly. In the homo-nonamer target T165/H1036 (A3B3C3), the first homo-trimer (A3) and one hetero-dimer (BC) were built based on available templates. Then, they were docked to form partial complexes (A3BC) and the full assembly was finally built by symmetry.

In the same line, target T170/H1060 was a challenging hetero-27mer, in which we applied an ad-hoc modeling procedure, also combining ab initio docking and template-based modeling. This assembly was formed by three rings with different composition and stoichiometry. The first ring was a homo-hexamer arranged as a dimer of trimers (2xA3) and was modelled by fitting two copies of the homo-trimeric x-ray structure (PDB 5NGJ) to available Cryo-EM data (EMDB ID: EMD-3689), followed by minimization. The second ring was formed by three subunits of one protein and twelve subunits of a second protein (B3C12) and was modelled using available monomeric models from CASP-hosted servers. Basically, the homo-trimer (B3) was modelled by building dimers (B2) with ab initio docking and generating the trimer by symmetry, followed by minimization and re-scoring by pyDock. The homo-trimer (B3) was docked to monomeric models of the second protein (C) obtained from CASP-hosted servers, and then B3C3 models were built by symmetry. In parallel, tetramers of the second protein (C4) were modelled by building dimers

112
(C2) with \textit{ab initio} docking and superimposing them on a tetrameric template (PDB 4BEG). The hetero-15meric ring (B3C12) was finally built by superimposing the homo-tetrameric models of the second protein (C4) onto each C subunit in the docking models (B3C3). The third homo-hexameric ring (D6) was modelled by superimposing the x-ray structure of the monomer (PDB 4JMQ) on available templates (PDB 4DIV and 2X8K), followed by minimization and pyDock scoring. The final assembly of the modelled rings was done with the help of \textit{ab initio} docking, selecting only models in which the symmetry axes of the rings were aligned. The same criteria was used in the scorers experiment.

**Availability**
The pyDock 3.0 program is available for academic use as a GNU/Linux binary and as a web server (https://life.bsc.es/pid/pydock/).