SOFTWARE TOOL ARTICLE

REVISED Fragger: a protein fragment picker for structural queries [version 2; referees: 2 approved]

Francois Berenger [1], David Simoncini[2], Arnout Voet[3], Rojan Shrestha[4], Kam Y.J. Zhang [5]

[1]System Cohort Division, Medical Institute of Bioregulation, Kyushu University, Fukuoka, Japan
[2]LISBP, Toulouse, France
[3]Laboratory of Biomolecular Modelling and Design, KU Leuven, Heverlee, Belgium
[4]Department of Systems and Computational Biology, Albert Einstein College of Medicine, Bronx, NY, USA
[5]Structural Bioinformatics Team, Division of Structural and Synthetic Biology, Center for Life Science Technologies, RIKEN, Yokohama, Kanagawa, Japan

v2 **First published:** 22 Sep 2017, **6**:1722 (doi: 10.12688/f1000research.12486.1)
Latest published: 10 Apr 2018, **6**:1722 (doi: 10.12688/f1000research.12486.2)

Abstract

Protein modeling and design activities often require querying the Protein Data Bank (PDB) with a structural fragment, possibly containing gaps. For some applications, it is preferable to work on a specific subset of the PDB or with unpublished structures. These requirements, along with specific user needs, motivated the creation of a new software to manage and query 3D protein fragments. Fragger is a protein fragment picker that allows protein fragment databases to be created and queried. All fragment lengths are supported and any set of PDB files can be used to create a database. Fragger can efficiently search a fragment database with a query fragment and a distance threshold. Matching fragments are ranked by distance to the query. The query fragment can have structural gaps and the allowed amino acid sequences matching a query can be constrained via a regular expression of one-letter amino acid codes. Fragger also incorporates a tool to compute the backbone RMSD of one versus many fragments in high throughput. Fragger should be useful for protein design, loop grafting and related structural bioinformatics tasks.

Open Peer Review

Referee Status: ✓ ✓

	Invited Referees	
	1	**2**
REVISED **version 2** published 10 Apr 2018		
version 1 published 22 Sep 2017	✓ report	✓ report

1 **Pierre Tuffery** , Paris Diderot University, France

2 **Charlotte Deane** , University of Oxford, UK
Saulo de Oliveira, Oxford University, UK

Discuss this article

Comments (0)

Corresponding author: Kam Y.J. Zhang (kamzhang@riken.jp)

Author roles: Berenger F: Conceptualization, Formal Analysis, Methodology, Software, Validation, Writing – Original Draft Preparation, Writing – Review & Editing; **Simoncini D**: Validation, Visualization, Writing – Review & Editing; **Voet A**: Validation, Writing – Review & Editing; **Shrestha R**: Validation, Writing – Review & Editing; **Zhang KYJ**: Conceptualization, Funding Acquisition, Resources, Supervision, Writing – Original Draft Preparation, Writing – Review & Editing

Competing interests: No competing interests were disclosed.

How to cite this article: Berenger F, Simoncini D, Voet A *et al.* **Fragger: a protein fragment picker for structural queries [version 2; referees: 2 approved]** *F1000Research* 2018, **6**:1722 (doi: 10.12688/f1000research.12486.2)

Grant information: This work was supported by the "Initiative Research Unit" program from RIKEN, Japan, the Japanese Society for the Promotion of Science (JSPS) and computing resources on the RIKEN Integrated Cluster of Clusters (RICC). FB is a JSPS international fellow.
The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

First published: 22 Sep 2017, **6**:1722 (doi: 10.12688/f1000research.12486.1)

REVISED **Amendments from Version 1**

Thanks to the referees' recommendations, we have improved our manuscript in this revised version. An extra paragraph was added in the introduction to describe the rationale underlying Fragger. Several citations to web servers and protein fragment databases were added to provide a broader overview of similar approaches. Some variables in the algorithm have been renamed to facilitate easy understanding. A new paragraph and one reference about choosing good reference fragments were added. Several changes were made to clarify the choice of reference fragments, and the impact of the effective cutoff values on search speed.

See referee reports

Introduction

Nowadays, a large number of protein structures are available (122,761 as of July 2017 at RCSB) and protein fragments are frequently used in structural bioinformatics. Protein structure prediction methods such as Rosetta[1], QUARK[2] and EdaFold[3,4] use protein fragments as building blocks. Protein fragments are also used in crystallographic phasing[5–7] and model rebuilding[8]. The quality of protein models can be improved by combining protein fragments with molecular dynamics[9]. Other applications include the curation of unresolved loops in crystal structures[10,11], grafting of loop sequences on protein scaffolds and other protein design algorithms[12,13].

When there are too many fragments to search from, an efficient strategy is necessary to reach sub-linear search times. This problem is well-known to the chemoinformatics community, which has developed several efficient strategies to screen large databases of small molecules. For example, geometric embedding and locality sensitive hashing[14], kd-trees[15], a tree data structure (called μ-tree) with a heuristic[16], bounds of similarity scores for chemical fingerprints[17] and a proximity filter based on the logical exclusive or operator[18] have all been developed to this end.

Currently, several fragment pickers[19–22] and protein fragment databases[23–28] are available. Of particular interest is the Super method[20] that uses the lower bound of RMSD[29] to screen the whole fragment space. However, our research on protein design and refinement of protein decoys for crystallographic phasing required specific options and therefore a new fragment picker.

Methods

Implementation

Algorithm 1. Query with a fragment and an RMSD threshold. Comments are enclosed between braces

```
Input: $D$: fragment set to query
Input: $R$: reference fragment set
Input: $q$: query fragment
Input: $d_q$: RMSD threshold
Output: $M$: matching fragment set
  $M \leftarrow D$
  {fuzzy query: prune the fragment space}
  for $r_j$ in $R$ do
    $d \leftarrow distance(q, r_j)$
    $d_{inf} \leftarrow d - d_q$
    $d_{sup} \leftarrow d + d_q$
    {$distance(f_i, r_j)$ comes from the database index}
    $M \leftarrow \{\forall f_i \in M \mid distance(f_i, r_j) \in [d_{inf}, d_{sup}]\}$
  end for
  {exact query: refine the result of pruning}
  $M \leftarrow \{\forall f_i \in M \mid distance(f_i, q) \leq d_q\}$
  return $M$
```

Fragger exploits the triangular inequality of RMSD[30] to prune the fragment space (Figure 1 and Algorithm 1). RMSDs are computed efficiently via the QCP method[31]. Fragger is written in OCaml[32], except backbone RMSD computations which are performed with a new version of the C++ ranker tool from Durandal[33]. Computations are parallelized on multi-core computers via the Parmap library[34].

Fragger allows a database to be queried with a fragment and an RMSD threshold. Matching fragments are ranked by RMSD to the query. Fragger's ranker tool allows to compute the backbone RMSD of a single fragment versus many. Fragger can deal with residue gaps or a selection of residues from the query, create a fragment database from a set of Protein Data Bank (PDB) files, work with all fragment lengths and extract specific or randomly-chosen fragments from a database.

Compared to existing fragment pickers, some of the specific functionalities required by users include:

- Outputing only the N best or N first found fragments matching a query (this can make a query terminate faster)

- Constraining the amino acid sequences allowed to match a query (for loop grafting; such filtering is applied after RMSD pruning of the fragment space)
- Reading and writing PDB fragments from/to a binary format (faster than reading/writing regular PDB files)
- Preventing a list of PDB codes from matching a query
- Automatically varying the RMSD threshold to the query until a given number of fragments is reached.

Operation

Users need to install OPAM and the pdbset command from CCP4 in order to use Fragger.

Details on how to install Fragger and usage examples are provided in the README file of the released software.

Results and discussion

Tests were performed on one core of a 2.4GHz Intel Xeon workstation with 12GB of RAM running Ubuntu Linux 12.04. The PDB dataset is composed of all proteins determined by X-ray, without highly similar sequences (30% sequence identity cutoff) in order to create a challenging set of fragments to benchmark a protein design algorithm. It contains 13,554 PDBs. PDBs were extracted from the protein databank website using the advanced search tab and ticking the "Retrieve only representatives at 30% sequence identity" box. Querying with a three (resp. nine) residues fragment takes at least 6.75s (resp. 5.2s).

Query times vary with the query fragment, reference fragments, indexed proteins and RMSD tolerance to the query. In general, the longer the required fragment length and the smaller the RMSD tolerance, the faster the query.

Reference fragments can be chosen randomly. Pruning of the search space is better if there are at least three reference fragments, far from each other. Once a RMSD index has been computed for a randomly chosen fragment (f_i), taking the furthest fragment from it (f_j) and the median fragment (f_k) would give three acceptable reference fragments. For interested contributors, some good heuristics can be found in the literature but were not implemented in Fragger, like Brin's greedy algorithm[35].

For one time tasks, it is not necessary to create RMSD indices and actually query a database, as fragments extraction and RMSD computations are fast enough. For example, it takes only 15s to generate all (41,200) fragments of 13 residues starting with alanine and ending with glycine (middle of Figure 1). Ranking them to the query takes 1.5s. When working on PDB files, the ranker tool included with Fragger can compute 66,580 (resp. 23,784) RMSD/s on the backbone of three (resp. nine) residue fragments. These numbers become 304,149 (resp. 138,744) RMSD/s when working on Fragger's binary-encoded PDBs. In the future, it might be possible to improve the performance of Fragger by incorporating a faster score than RMSD, such as BCscore[36].

Fragger can be useful for protein design, loop grafting and retrieval of candidates to rebuild low-confidence regions of protein models[6].

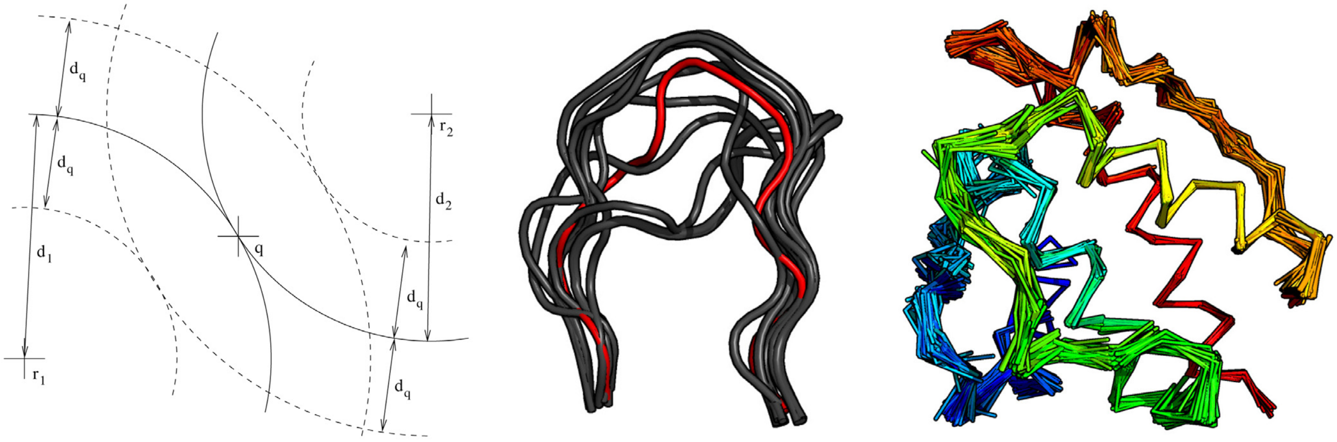

Figure 1. Left: pruning the fragment space for query distance d_q and query fragment q. q is at distance d_1 (resp. d_2) from reference fragment r_1 (resp. r_2). Only fragments which are both within $d_1 \pm d_q$ of r_1 and $d_2 \pm d_q$ of r_2 will undergo an RMSD calculation. Middle: 13 residues loops that can connect residue ALA 98 to GLY 110 in chain A of PDB 1MEL. The query loop is shown in red. Only its first and last three residues were used to rank the retrieved fragments. Right: Backbone of PDB 1BKR covered with ten residue fragments from non-homologous proteins retrieved with Fragger.

Data availability

All data underlying the results are available as part of the article and no additional source data are required.

Software availability

Fragger can be downloaded from: https://github.com/UnixJunkie/fragger

Archived source code at the time of publication: https://zenodo.org/record/877320

Software license: LGPL.

Competing interests

No competing interests were disclosed.

Grant information

This work was supported by the "Initiative Research Unit" program from RIKEN, Japan, the Japanese Society for the Promotion of Science (JSPS) and computing resources on the RIKEN Integrated Cluster of Clusters (RICC). FB is a JSPS international fellow.

The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

References

1. Leaver-Fay A, Tyka M, Lewis SM, *et al.*: **Rosetta3: An Object-Oriented Software Suite for the Simulation and Design of Macromolecules**. *Methods Enzymol.* Academic Press, 2011; **487**: 545–574.
PubMed Abstract | Publisher Full Text | Free Full Text
2. Xu D, Zhang Y: ***Ab initio* protein structure assembly using continuous structure fragments and optimized knowledge-based force field.** *Proteins.* 2012; **80**(7): 1715–1735.
PubMed Abstract | Publisher Full Text | Free Full Text
3. Simoncini D, Berenger F, Shrestha R, *et al.*: **A Probabilistic Fragment-Based Protein Structure Prediction Algorithm.** *PLoS One.* 2012; **7**(7): e38799.
PubMed Abstract | Publisher Full Text | Free Full Text
4. Simoncini D, Schiex T, Zhang KY: **Balancing exploration and exploitation in population-based sampling improves fragment-based *de novo* protein structure prediction.** *Proteins.* 2017; **85**(5): 852–858.
PubMed Abstract | Publisher Full Text
5. Rodriguez DD, Grosse C, Himmel S, *et al.*: **Crystallographic *ab initio* protein structure solution below atomic resolution.** *Nat Methods.* 2009; **6**(9): 651–653.
PubMed Abstract | Publisher Full Text
6. Shrestha R, Simoncini D, Zhang KY: **Error-estimation-guided rebuilding of *de novo* models increases the success rate of *ab initio* phasing.** *Acta Crystallogr D Biol Crystallogr.* 2012; **68**(Pt 11): 1522–1534.
PubMed Abstract | Publisher Full Text
7. Shrestha R, Zhang KY: **A fragmentation and reassembly method for *ab initio* phasing.** *Acta Crystallogr D Biol Crystallogr.* 2015; **71**(Pt 2): 304–312.
PubMed Abstract | Publisher Full Text
8. Adams PD, Afonine PV, Bunkóczi G, *et al.*: ***PHENIX*: a comprehensive Python-based system for macromolecular structure solution.** *Acta Crystallogr D Biol Crystallogr.* 2010; **66**(Pt 2): 213–221.
PubMed Abstract | Publisher Full Text | Free Full Text
9. Zhang J, Liang Y, Zhang Y, *et al.*: **Atomic-Level Protein Structure Refinement Using Fragment-Guided Molecular Dynamics Conformation Sampling.** *Structure.* 2011; **19**(12): 1784–1795.
PubMed Abstract | Publisher Full Text | Free Full Text
10. Lee J, Lee D, Park H, *et al.*: **Protein loop modeling by using fragment assembly and analytical loop closure.** *Proteins.* 2010; **78**(16): 3428–36.
PubMed Abstract | Publisher Full Text | Free Full Text
11. Shehu A, Clementi C, Kavraki LE, *et al.*: **Modeling protein conformational ensembles: from missing loops to equilibrium fluctuations.** *Proteins.* 2006; **65**(1): 164–79.
PubMed Abstract | Publisher Full Text
12. Claessens M, Van Cutsem E, Lasters I, *et al.*: **Modelling the polypeptide backbone with 'spare parts' from known protein structures.** *Protein Eng.* 1989; **2**(5): 335–45.
PubMed Abstract | Publisher Full Text
13. Tsai HH, Tsai CJ, Ma B, *et al.*: ***In silico* protein design by combinatorial assembly of protein building blocks.** *Protein Sci.* 2004; **13**(10): 2753–65.
PubMed Abstract | Publisher Full Text
14. Cao Y, Jiang T, Girke T: **Accelerated similarity searching and clustering of large compound sets by geometric embedding and locality sensitive hashing.** *Bioinformatics.* 2010; **26**(7): 953–959.
PubMed Abstract | Publisher Full Text | Free Full Text
15. Agrafiotis DK, Lobanov VS: **An efficient implementation of distance-based diversity measures based on *k*-d trees.** *J Chem Inf Comput Sci.* 1999; **39**(1): 51–58.
Publisher Full Text
16. Xu H, Agrafiotis DK: **Nearest neighbor search in general metric spaces using a tree data structure with a simple heuristic.** *J Chem Inf Comput Sci.* 2003; **43**(6): 1933–1941.
PubMed Abstract | Publisher Full Text
17. Swamidass SJ, Baldi P: **Bounds and algorithms for fast exact searches of chemical fingerprints in linear and sublinear time.** *J Chem Inf Model.* 2007; **47**(2): 302–317.
PubMed Abstract | Publisher Full Text | Free Full Text
18. Baldi P, Hirschberg DS, Nasr RJ: **Speeding up chemical database searches using a proximity filter based on the logical exclusive or.** *J Chem Inf Model.* 2008; **48**(7): 1367–1378.
PubMed Abstract | Publisher Full Text
19. Gront D, Kulp DW, Vernon RM, *et al.*: **Generalized fragment picking in Rosetta: design, protocols and applications.** *PLoS One.* 2011; **6**(8): e23294.
PubMed Abstract | Publisher Full Text | Free Full Text
20. Collier JH, Lesk AM, Garcia de la Banda M, *et al.*: **Super: a web server to rapidly screen superposable oligopeptide fragments from the protein data bank.** *Nucleic Acids Res.* 2012; 40(Web Server issue): W334–W339.
PubMed Abstract | Publisher Full Text | Free Full Text
21. Guyon F, Martz F, Vavrusa M, *et al.*: **BCSearch: fast structural fragment mining over large collections of protein structures.** *Nucleic Acids Res.* 2015; **43**(W1): W378–W382.
PubMed Abstract | Publisher Full Text | Free Full Text
22. Santos KB, Trevizani R, Custodio FL, *et al.*: **Profrager web server: Fragment libraries generation for protein structure prediction**. In *Proceedings of the International Conference on Bioinformatics & Computational Biology (BIOCOMP).* The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2015; 38.
Reference Source
23. Kim DE, Chivian D, Baker D: **Protein structure prediction and analysis using the Robetta server.** *Nucleic Acids Res.* 2004; **32**(Web Server issue): W526–W531.
PubMed Abstract | Publisher Full Text | Free Full Text
24. Samson AO, Levitt M: **Protein segment finder: an online search engine for segment motifs in the pdb.** *Nucleic Acids Res.* 2009; **37**(Database issue): D224–D228.
PubMed Abstract | Publisher Full Text | Free Full Text
25. Debret G, Martel A, Cuniasse P: **RASMOT-3D PRO: a 3D motif search webserver.** *Nucleic Acids Res.* 2009; **37**(Web Server issue): W459–W464.
PubMed Abstract | Publisher Full Text | Free Full Text
26. Vanhee P, Verschueren E, Baeten L, *et al.*: **BriX: a database of protein building blocks for structural analysis, modeling and design.** *Nucleic Acids Res.* 2011; **39**(Database issue): D435–D442.
PubMed Abstract | Publisher Full Text | Free Full Text
27. Nagarajan R, Siva Balan S, Sabarinathan R, *et al.*: ***Fragment Finder 2.0*: a computing server to identify structurally similar fragments.** *J Appl Cryst.* 2012; **45**(2): 332–334.
Publisher Full Text
28. Budowski-Tal I, Nov Y, Kolodny R: **FragBag, an accurate representation of protein structure, retrieves structural neighbors from the entire PDB quickly and accurately.** *Proc Natl Acad Sci U S A.* 2010; **107**(8): 3481–3486.
PubMed Abstract | Publisher Full Text | Free Full Text

29. Tramontano A, Lesk AM: **Common features of the conformations of antigen-binding loops in immunoglobulins and application to modeling loop conformations.** *Proteins.* 1992; **13**(3): 231–245.
PubMed Abstract | Publisher Full Text

30. Steipe B: **A revised proof of the metric properties of optimally superimposed vector sets.** *Acta Crystallogr A.* 2002; **58**(Pt 5): 506.
PubMed Abstract | Publisher Full Text

31. Theobald DL: **Rapid calculation of RMSDs using a quaternion-based characteristic polynomial.** *Acta Crystallogr A.* 2005; **61**(Pt 4): 478–480.
PubMed Abstract | Publisher Full Text

32. Leroy X, Doligez D, Frisch A, *et al.*: **The OCaml system release 4.00 Documentation and user's manual**. INRIA, France, 2012.
Reference Source

33. Berenger F, Shrestha R, Zhou Y, *et al.*: **Durandal: fast exact clustering of protein decoys.** *J Comput Chem.* 2012; **33**(4): 471–474.
PubMed Abstract | Publisher Full Text

34. Daneluttoa M, Di Cosmo R: **A "Minimal Disruption" Skeleton Experiment: Seamless Map and Reduce Embedding in OCaml.** *Procedia Comput Sci.* 2012; **9**: 1837–1846.
Publisher Full Text

35. Brin S: **Near neighbor search in large metric spaces**. In *Proceedings of the 21th International Conference on Very Large Data Bases.* VLDB '95, San Francisco, CA USA, Morgan Kaufmann Publishers Inc. 1995; 574–584.
Reference Source

36. Guyon F, Tufféry P: **Fast protein fragment similarity scoring using a Binet-Cauchy kernel.** *Bioinformatics.* 2014; **30**(6): 784–791.
PubMed Abstract | Publisher Full Text

Open Peer Review

Current Referee Status: ✓ ✓

Version 1

Referee Report 18 January 2018

doi:10.5256/f1000research.13520.r28334

✓ **Charlotte Deane** [1], **Saulo de Oliveira** [2]

[1] Department of Statistics, University of Oxford, Oxford, UK

[2] Department of Statistics, Oxford University, Oxford, UK

In this paper, the authors describe Fragger, a web server for retrieving fragments from a database of structures based on a query fragment. Their program allows for customisation in terms of number of fragments output and in terms of sequence constraints.

The paper is made up of previously published methods put together as a potentially useful package.

The Super method (mentioned in the introduction of the manuscript) seems to fulfil the same purpose as Fragger. No comparison is provided between the two methods in terms of performance and quality of fragments output.

It is unclear from the Methods section how the reference fragment set is selected. It is mentioned in the Results that these fragments can be selected at random, but the authors never discuss whether this choice can have an impact on the performance of the algorithm.

Is the rationale for developing the new software tool clearly explained?
Partly

Is the description of the software tool technically sound?
Yes

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?
Partly

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?
Yes

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?
Partly

Competing Interests: No competing interests were disclosed.

We have read this submission. We believe that we have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Author Response 04 Apr 2018

Kam Zhang, RIKEN, Japan

> The Super method (mentioned in the introduction of the manuscript) seems
> to fulfil the same purpose as Fragger. No comparison is provided between
> the two methods in terms of performance and quality of fragments output.

Super uses only CA to calculate RMSD.
Fragger uses backbone atoms to calculate RMSD, since our users want
to preserve secondary structure information.
Since Super uses four times fewer atoms than Fragger for each RMSD calculation,we don't think
such a comparison would be fair.

> It is unclear from the Methods section how the reference fragment set
> is selected. It is mentioned in the Results that these fragments can be
> selected at random, but the authors never discuss whether this choice
> can have an impact on the performance of the algorithm.

Indeed, good reference fragments can improve the search performance.
We have added a new paragraph and one reference about choosing
good reference fragments.

Competing Interests: No competing interests were disclosed.

Referee Report 11 October 2017

doi:10.5256/f1000research.13520.r26264

✓

Pierre Tuffery

Molécules Thérapeutiques In Silico (MTi) (UMR-S 973), French National Institute of Health and Medical Research (INSERM), Sorbonne Paris Cité, Paris Diderot University, Paris, France

This paper describes an approach to quickly scan large collection of proteins to identify fragments similar to a request. Not considering indels, this approach is, as stated by the authors, in the context of fragment grafting, loop modeling, protein design or crystallographic phasing.

The metrics used to quantify the similarity is that of the RMSd.
The rationale here is to to use the triangular inequality of RMSd to setup a two step procedure:

- decompose the complete set of fragments present in the collection of proteins by as a limited subset of representative fragments

- quickly identify the representative fragments similar to the query in a way to perform effective pruning of the complete collection of fragments, ensuring not discarding the matching fragments, and then perform a systematic search for the fragments of the classes associated with the matching representative fragments.

This kind of approach has been used in several contexts and is interesting. The manuscript however could easily be improved.

Here are some specific comments:

- The introduction could benefit from a better description of the rationale underlying Fragger, including its use in different contexts. For instance, such a strategy has also been used for the fast similarity search of small compounds.

- The introduction could benefit from a larger overview of the approaches that have been setup to address questions similar to that of Fragger. There are also a series of web servers focusing on this goal that are not cited.

- The way the algorithm is described makes it rather uneasy to understand. There could first be some awkwardness in the notations. For instance, in the algorithm description, blanks between d and b and between r and f could be discarded. Secondly, it could be difficult for a reader to understand the role of the representative fragments, the way they are identified and used from the present description of the algorithm. Probably an additional flowchart or figure to explain it would be welcome.

- The critical parameters of the procedure are not really identified. What are the effective cutoff values, how do they impact on the search ?

- It seems Fragger offers possibilities to constrain amino acidd sequences. Is it a prior or a posterior filtering ?

Is the rationale for developing the new software tool clearly explained?
Partly

Is the description of the software tool technically sound?
Partly

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?
Yes

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?
Yes

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?
Partly

Competing Interests: No competing interests were disclosed.

I have read this submission. I believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Author Response 04 Apr 2018

Kam Zhang, RIKEN, Japan

> - The introduction could benefit from a better description of the
> rationale underlying Fragger, including its use in different contexts. For
> instance, such a strategy has also been used for the fast similarity
> search of small compounds.

We have added an extra paragraph in the introduction to mention related methods found in chemoinformatics.

> - The introduction could benefit from a larger overview of the
> approaches that have been setup to address questions similar to that of
> Fragger. There are also a series of web servers focusing on this goal
> that are not cited.

We have added several citations to web servers and protein fragment databases which are using various methods.

> - The way the algorithm is described makes it rather uneasy to
> understand. There could first be some awkwardness in the notations. For
> instance, in the algorithm description, blanks between d and b and
> between r and f could be discarded. Secondly, it could be difficult
> for a reader to understand the role of the representative fragments,
> the way they are identified and used from the present description of
> the algorithm. Probably an additional flowchart or figure to explain it
> would be welcome.

We have renamed some variables in the algorithm to bypass typographic problems introduced by the journal's style-sheet.

We have also added a new paragraph and one reference about choosing good reference fragments.

> - The critical parameters of the procedure are not really identified. What
> are the effective cutoff values, how do they impact on the search ?

This is quite complex: the search speed is influenced by the protein database, the fragment length and the query RMSD tolerance.
In the manuscript, we now summarize the general trend as:
"In general, the longer the required fragment length and the smaller the RMSD tolerance, the faster the query.".

> - It seems Fragger offers possibilities to constrain amino acidd
> sequences. Is it a prior or a posterior filtering?

We updated the manuscript to indicate that this is done after geometric filtering.

Competing Interests: No competing interests were disclosed.