Supplementary Material

TITLE

SonicParanoid: fast, accurate, and easy orthology inference

AUTHORS

Salvatore Cosentino\textsuperscript{1,*} and Wataru Iwasaki\textsuperscript{1,2,3,*}

\textsuperscript{1} Department of Biological Sciences, Graduate School of Science, The University of Tokyo, Bunkyo-ku, Tokyo 113-0032, Japan

\textsuperscript{2} Department of Computational Biology and Medical Sciences, Graduate School of Frontier Sciences, The University of Tokyo, Kashiwa, Chiba 277-8568, Japan

\textsuperscript{3} Atmosphere and Ocean Research Institute, The University of Tokyo, Kashiwa, Chiba 277-8564, Japan

*Correspondence to: salvocos@bs.s.u-tokyo.ac.jp (S.C.); iwasaki@bs.s.u-tokyo.ac.jp (W.I.)
Supplementary Text

Orthology Inference in SonicParanoid

SonicParanoid is a graph-based orthology inference tool. Given $N$ input proteomes, SonicParanoid conducts all-vs-all protein alignment for $N \times (N - 1)$ between-proteome and $N$ within-proteome pairs using MMseqs2 (Steinegger and Söding, 2017).

The algorithm used in InParanoid (Remm et al., 2001) serves as a backbone for the orthology inference in SonicParanoid, whereas changes to the core algorithm reduce the execution time and increase the usability. To reduce the execution time, the within-proteome all-vs-all alignments are reused and all second-pass alignments are skipped. SonicParanoid also avoids the bootstrapping step that is performed by InParanoid by default, because it requires additional sequence alignments that are considerably time consuming despite the fact that the previously computed confidence scores are not modified by the bootstrap. Other technical improvements that reduced the execution time include the use of small and simplified alignment files. For example, the processed within-proteome alignment file for *Trichomonas vaginalis* produced by InParanoid is approximately 1.8 GB, while the one created by SonicParanoid is approximately 380 MB. Furthermore, SonicParanoid introduces a configurable threshold for the sequence length differences and a new scoring function.

Regarding the length-difference threshold, in addition to the sequence-overlap check (*Overlap criteria*) performed by InParanoid (Östlund et al., 2010), we introduced a configurable threshold for directly checking the sequence length differences. The default value for the length-difference threshold in SonicParanoid is set to 0.5 based on the results of the generalized species
tree discordance tests at the last universal common ancestor (LUCA) level in the sensitive mode (Supplementary Figures S3, S11, and S12). The test based on the LUCA dataset (Altenhoff et al., 2016) was used as a reference because it uses the majority of the predicted orthologs.

Regarding the new scoring function, let \( a \) and \( b \) be orthologs between proteomes A and B, \( a_p \) a candidate in-paralog in proteome A, and \( \text{score}(x, y) \) the BLAST bitscore between proteins \( x \) and \( y \). The original InParanoid algorithm identifies in-paralogs using the following scoring function

\[
f(a_p) = \frac{\text{score}(a, a_p) - \text{score}(a, b))}{(\text{score}(a, a) - \text{score}(a, b))}
\]

InParanoid ver 4.1 (and SonicParanoid) predicts only proteins with confidence scores above 0.05 as in-paralogs. Alternatively, SonicParanoid uses a new scoring function that penalizes candidate in-paralogs considerably longer or shorter than the main orthologs (Supplementary Figure S2b, left plot). By letting \( l \) and \( s \) be the lengths of the longer and shorter sequence between \( a_p \) and \( a \), respectively, \( d \) the length difference between \( l \) and \( s \), and \( ldr \) the length difference ratio between \( a_p \) and \( a \) (i.e., \( d / l \)), SonicParanoid identifies in-paralogs using the following function:

\[
f(a_p) = \begin{cases} 
\frac{\text{score}(a, a_p) - \text{score}(a, b)}{\text{score}(a, a) - \text{score}(a, b)}, & \text{if } d = 0 \\
\left( \frac{\text{score}(a, a_p) - \text{score}(a, b)}{\text{score}(a, a) - \text{score}(a, b)} \right) \left( 1 - \left( \frac{3}{2} \right)^{\frac{ldr-1}{ldr}} \right), & \text{otherwise}
\end{cases}
\]

which applies penalties that grow exponentially with the sequence length difference between candidate in-paralogs and main orthologs (Supplementary Figure S2b, left plot).

Another difference with the InParanoid algorithm is in the way that overlapping groups of orthologs are merged and removed. In InParanoid, overlapping groups are clustered in the order of their similarity scores, beginning from the most similar groups. Groups are then merged,
separated, or removed based on the comparisons of the confidence scores of the main orthologs (identified using bidirectional-best-hit) in each group. This process can be time consuming for big ortholog tables and uses an arbitrary confidence score threshold of 0.5 to merge groups. In contrast, SonicParanoid uses a greedy approach that treats groups as elements of numerical sets and iteratively merges them, by starting from the groups with the highest cumulative alignment scores (Supplementary Figure S4). The merge-and-remove step in SonicParanoid relies on the cumulative alignment scores of groups and avoids the use of thresholds based on the confidence score, which could be inadequate when comparing paralogs of organisms with different evolutionary distances.

The multi-species ortholog groups are estimated using a modified version of QuickParanoid [http://pl.postech.ac.kr/QuickParanoid], which is a C implementation of MultiParanoid and was tested to work on 273 proteomes from the InParanoid8 database (Sonnhammer and Östlund, 2015). While the original QuickParanoid (and MultiParanoid) requires users to collect ortholog relationship files and create extra configuration files for themselves, SonicParanoid automates the whole process, which can be error-prone and cumbersome if manually conducted (Supplementary Figure S1).

**All-vs-all Alignments and Memory Requirements**

SonicParanoid uses MMseqs2 (Steinegger and Söding, 2017) (release 1-c7a89, commit c552cce) for all-vs-all protein alignments, which computes local alignments with a level of sensitivity comparable or higher to that of BLAST, but runs faster on computers with multiple CPUs, which are commonly available today. The ‘min-ungapped-score’ parameter of MMseqs2 is set to 30.
For the *fast, default, and sensitive* modes, the ‘-s’ parameter (a sensitivity threshold) is set to 2.5, 4.0, and 6.0, respectively (Supplementary Table S1). Aside from these two parameters, the default settings are used.

The reduced execution time of MMseqs2 is achieved at the cost of large memory usage especially when complete protein databases (e.g., NCBI nr) are used. To make sure that MMseqs2 does not require an amount of memory that could negatively affect the usability of SonicParanoid, we monitored the memory usage of SonicParanoid during the processing and post-processing of the alignments for the QfO dataset. The memory usage analysis using mprof [https://pypi.org/project/memory_profiler] showed an average system memory usage of 6.5 Gigabytes and a maximum of 10.4 Gigabytes, when eight CPUs are used (Supplementary Figure S8; note that SonicParanoid is based on all-vs-all alignments of proteome-pairs). Although the amount of memory used is relatively large when compared to those of other alignment tools (for example, that of DIAMOND), such an amount is acceptable for most computers optimized for bioinformatics analyses. When 4 CPUs are used, the average memory usage becomes less than 4 gigabytes, which is acceptable for typical modern computers.

SonicParanoid makes MMseqs2 generate a relatively large index file for each input file (approximately 1.2 gigabytes for each proteome) but deletes it at the end of the execution. In addition, SonicParanoid has an option to avoid the creation of the MMseqs2 index files, although this option increases the processing time for the alignment step by approximately 5~10% depending on the hardware. Furthermore, SonicParanoid automatically avoids the creation of the MMseqs2 index files if the available disk space is smaller than that required to store them and informs the users through a warning message in the command line.
Scalability Assessment of Multi-species Ortholog Group Estimation

To assess the speed and scalability of SonicParanoid in the estimation of multi-species ortholog groups, we performed multi-species orthology inference using SonicParanoid and MultiParanoid on 27 datasets of different sizes constructed using ortholog relationship files obtained from the InParanoid 8 database (Sonnhammer & Östlund 2015). For the smallest dataset, that was composed of 45 ortholog relationship files inferred for ten proteomes, SonicParanoid and MultiParanoid required four seconds and 26 minutes, respectively (i.e., 380X faster) (Supplementary figure S7). For the datasets with ortholog relationships from twenty or more proteomes, while MultiParanoid was unable to complete the computation, SonicParanoid was remarkably fast and processed datasets with ortholog relationships from up to 60 eukaryotes in about one minute, and the complete InParanoid 8 dataset (37,128 ortholog relationship files computed from 273 proteomes) in approximately 27 minutes.

Speed Evaluation

The QfO proteome dataset (Gabaldón et al., 2009) was downloaded in July 2016 from ftp://ftp.ebi.ac.uk/pub/databases/reference_proteomes/previous_releases/qfo_release-2011_04. To comply with the data used in the benchmarks described in Altenhoff et al. 2016, we used the 2011 version of the dataset, which was composed of 66 proteomes (40 eukaryotes and 26 prokaryotes) (Supplementary Figure S5). The execution time of SonicParanoid was compared to those of InParanoid (ver. 4.1), Proteinortho (ver. 5.15) (Lechner et al., 2011), and OrthoFinder (ver. 2.1.2) (Emms and Kelly, 2015). InParanoid used legacy-BLAST (BLASTALL 2.2.18)
(Altschul et al., 1997) for the all-vs-all alignments. Proteinortho and OrthoFinder used BLASTP (Camacho et al., 2009) (ver. 2.4.0+). OrthoFinder also used DIAMOND (ver. 0.9.10) (Buchfink et al., 2015).

The tests to compare the execution time of SonicParanoid, InParanoid, OrthoFinder, and Proteinortho were performed on a RedHat Linux 4.8.5 server with 40 Intel Xeon CPUs at 2.6 GHz, 500 GB of memory (no limit was set on the memory usage), and one Intel P3700 solid-state disk. The total CPU-time was measured using the perf performance analyzing tool for Linux, using the command ‘perf stat -e cpu-clock, task-clock, instruction’ followed by the program to be executed. For each tool eight CPUs were used. It should be noted that SonicParanoid, OrthoFinder, and Proteinortho automatically parallelize one-CPU processes when performing the all-vs-all alignments. For InParanoid, eight instances each of which used one CPU were run in parallel by the authors (for approximately three months).

**Accuracy Evaluation**

The accuracy benchmark was performed by uploading the ortholog relationship files obtained for the QfO dataset to the orthology benchmarking service (Altenhoff et al., 2016). For the assessment of SonicParanoid, the 7.8 million ortholog relationships inferred for the complete QfO proteome dataset were fed into the benchmark. For Proteinortho, the 2.8 million ortholog relationships, which were much fewer than those inferred by SonicParanoid, were fed into the benchmark. For the assessment of InParanoid, OrthoFinder, and the remaining 10 tools, the evaluation results available at the benchmarking website were referenced.

The uploaded files were simple text files with two tab-separated columns of orthologous
proteins represented by their IDs. For Proteinortho, the orthologous pairs were extracted from the file `myproject.proteinortho-graph` generated at the end of each run. The uploaded ortholog relationships and results are available at the Orthology Benchmarking webpage 
[http://orthology.benchmarkservice.org/cgi-bin/gateway.pl?f=ShowProject]. SonicParanoid was benchmarked using several sequence length difference thresholds (no filtering, 25%, 50%, and 75%-length difference allowed) and the original InParanoid scoring function, thus a total of 15 datasets were uploaded to the benchmarking service (Supplementary Figures S11 and S12). The results shown in Figure 1b and Supplementary Figures S9 and S10 were obtained with an allowed maximum length difference ratio of 0.5, which is default in SonicParanoid.

**Usage**

To run SonicParanoid, users only need to execute a program, `sonicparanoid`, with the path to the directory containing the input proteomes as the only parameter. At the end of the execution, the output directory contains subdirectories (for each proteome pair) and a directory with a multi-species ortholog table file. Each subdirectory contains two ortholog relationship files (one in plain text and the other in the SQL format), while the intermediate alignment files are stored in the ‘alignments’ directory.

The *update* option facilitates data addition to and removal from a previous execution of SonicParanoid (Supplementary Figure S1). If users specify an output directory from a previous execution, the program automatically skips computation that was already completed. The *multi-species-only* option enables the inference of multi-species ortholog groups for subsets of proteomes by reusing the results of a previous SonicParanoid run; thus, users can easily obtain
detailed information about ortholog relationships among the subsets of organisms that they are studying. Because all required all-vs-all alignment should be already available in this case, the computation is quickly completed.

To meet various needs from quick assessment to detailed analysis, SonicParanoid provides fast, default, and sensitive modes that employ different filtering thresholds for the alignment process. While the default mode would suit most studies, the fast and sensitive modes may be used for evolutionarily close and distant species, respectively (Supplementary Figures S9 and S11). Finally, SonicParanoid can halt and restart the execution of a run without losing previously computed results, which may be useful, for example, to users who perform orthology inference using their laptops.

Implementation, Software, and Hardware Requirements and Availability.

SonicParanoid was implemented using the Python programming language and its extension, Cython (Behnel et al., 2011). The software requirements include the installation of Python (version 3.5 or above), the Cython, Numpy, and Pandas modules, and MMseqs2 (Steinegger and Söding, 2017) (release 1-c7a89, commit c552cce). SonicParanoid was tested on computers running Linux and Mac OSX and should work on any UNIX-based operative system. The suggested minimum hardware requirements include a multi-core (at least four) CPU and eight Gigabytes of memory.

The official webpage of SonicParanoid can be found at http://iwasakilab.bs.s.u-tokyo.ac.jp/sonicparanoid. The software is available for download from the Python Package Index (https://pypi.org/project/sonicparanoid) and BitBucket.
(https://bitbucket.org/salvocos/sonicparanoid). SonicParanoid is free for academic use and is licensed under the GNU GENERAL PUBLIC LICENSE, Version 3 (GPLv3) (https://www.gnu.org/licenses/gpl-3.0.en.html).
Supplementary Fig. S1. Flowchart of SonicParanoid.
Supplementary Fig. S2. Effect of the new scoring function on confidence scores. Each data point represents a protein that was predicted as an in-paralog (green points) or rejected (red points) when SonicParanoid was applied to the QfO dataset. The x-axes show length difference folds (a) and ldr (b) to the main corresponding orthologs. The y-axes show confidence scores with (left) or without (right) the penalty. In SonicParanoid (same as in InParanoid), candidate in-paralogs with confidence scores below 0.05 are rejected. The left plot in (a) shows how proteins
considerably longer or shorter than the main orthologs are rejected through the application of the penalty, while these can be predicted as in-paralogs when the length-difference-based penalty is not applied (right plot in (a)). The left plot in (b) shows that the SonicParanoid function does not apply strong penalties to proteins with an ldr smaller than 0.05, allowing small length differences.
Supplementary Fig. S3. Numbers of predicted ortholog relationships using different in-paralog filtering thresholds and scoring functions. SonicParanoid was run in the sensitive (top), default (middle), and fast (bottom) modes using the SonicParanoid scoring function (blue) and the original InParanoid scoring function (red). The number on the top of each bar represents the percentage of rejected ortholog relationships in relation to the total amount of predicted relationships without any sequence-length difference filtering.
**Supplementary Fig. S4. Merge-and-remove step in SonicParanoid.** (a) Example of overlapping clusters of homologs with description. (b) Description of the greedy algorithm used to merge and remove clusters in SonicParanoid.
Supplementary Fig. S5. Composition of the QfO dataset. The left panel shows the numbers of proteomes ($N$) and proteome pairs ($N \times (N - 1) / 2$) for the entire QfO dataset and two taxonomy-based subsets. The right panel shows the average proteome sizes with standard deviations.
Supplementary Fig. S6. Speed of SonicParanoid on the complete, eukaryotic, and prokaryotic QfO datasets. Speed-up folds of SonicParanoid (in the fast, default, and sensitive modes) in relation to InParanoid, OrthoFinder2 (on BLASTP and DIAMOND), and Proteinortho are shown (Left: complete execution; right: orthology inference step only). The numbers in the square brackets represent the execution time in hours (left) or minutes (right). Eight processors were used by each tool.
Supplementary Fig. S7. Execution time and scalability of SonicParanoid in the inference of ortholog groups. SonicParanoid was used to infer ortholog groups from the ortholog relationship files in the InParanoid 8 database using a single CPU. An increasing number of proteomes were fed to SonicParanoid.
Supplementary Fig. S8. Memory use of SonicParanoid during all-vs-all alignments. The cumulative memory usage of SonicParanoid and DIAMOND was monitored using mprof [https://pypi.org/project/memory_profiler] during the processing of the alignments of the 66 proteomes in the QfO dataset. Eight CPUs were used for each tool. Due to the computational overhead caused by the memory monitoring tool, the execution times were increased.
Supplementary Fig. S9. Accuracy of SonicParanoid and other 13 methods based on species tree discordance tests. The results of (a) species tree discordance and (b) generalized species tree discordance tests are shown. Error bars indicate 95% confidence intervals.
Supplementary Fig. S10. Accuracy of SonicParanoid and other 13 methods based on reference gene tree and functional conservation tests. The results on (a) reference gene tree tests and (b) functional conservation tests are shown. Error bars indicate 95% confidence intervals.
Supplementary Fig. S11. Comparison of the accuracy of SonicParanoid using different in-paralog filtering thresholds and scoring functions. The results of (a) species tree discordance and (b) generalized species tree discordance tests are shown. Error bars indicate 95% confidence intervals.
Supplementary Fig. S12. Comparison of the accuracy of SonicParanoid using different in-paralog filtering thresholds and scoring functions. The results of (a) gene tree and (b) functional conservation tests are shown. Error bars indicate 95% confidence intervals.
Supplementary Table S1. Effects of the ‘-s’ parameter of MMseqs2 on the execution time and numbers of estimated ortholog relationships and groups. SonicParanoid was executed with the QfO dataset as input. When the sensitivity of MMseqs2 decreases, SonicParanoid becomes faster but fails to identify more ortholog relationships. We chose $s = 2.5, 4.0, \text{ and } 6.0$ for the fast, default, and sensitive modes, respectively.
REFERENCES

Altenhoff, A.M. et al. (2016) Standardized benchmarking in the quest for orthologs. *Nat. Methods*, **13**, 425–430.

Altschul, S.F. et al. (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.*, **25**, 3389–3402.

Behnel, S. et al. (2011) Cython: The Best of Both Worlds. *Comput. Sci. Eng.*, **13**, 31–39.

Buchfink, B. et al. (2015) Fast and sensitive protein alignment using DIAMOND. *Nat. Methods*, **12**, 59–60.

Camacho, C. et al. (2009) BLAST+: architecture and applications. *BMC Bioinformatics*, **10**, 421.

Emms, D.M. and Kelly, S. (2015) OrthoFinder: solving fundamental biases in whole genome comparisons dramatically improves orthogroup inference accuracy. *Genome Biol.*, **16**, 157.

Gabaldón, T. et al. (2009) Joining forces in the quest for orthologs. *Genome Biol.*, **10**, 403.

Lechner, M. et al. (2011) Proteinortho: detection of (co-)orthologs in large-scale analysis. *BMC Bioinformatics*, **12**, 124.

Ostlund, G. et al. (2010) InParanoid 7: new algorithms and tools for eukaryotic orthology analysis. *Nucleic Acids Res.*, **38**, D196–203.

Remm, M. et al. (2001) Automatic clustering of orthologs and in-paralogs from pairwise species comparisons. *J. Mol. Biol.*, **314**, 1041–1052.

Sonnhammer, E.L.L. and Östlund, G. (2015) InParanoid 8: orthology analysis between 273 proteomes, mostly eukaryotic. *Nucleic Acids Res.*, **43**, D234–9.

Steinegger, M. and Söding, J. (2017) MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nat. Biotechnol.*, **35**, 1026–1028.