Bacterial Identifier: Accelerating Bacterial Genome Detection

1st Julie Nejezchlebová
Department of Biomedical Engineering
Faculty of Electrical Engineering and Communication,
Brno University of Technology
Brno, Czech Republic
xnejez10vutbr.cz

2nd Ivan Rychlík
Department of Microbiology and Antimicrobial Resistance, Veterinary Research Institute,
Brno, Czech Republic
ivan.rychlik@vri.cz

3rd Jana Schwarzerová
Department of Biomedical Engineering
Faculty of Electrical Engineering and Communication,
Brno University of Technology
Brno, Czech Republic
Department of Molecular and Clinical Pathology and Medical Genetics, University Hospital Ostrava, Ostrava, Czech Republic
Molecular System Biology (MOSYS), University of Vienna,
Vienna, Austria
Jana.Schwarzerova@vut.cz

***Abstract* — Bacterial identification is crucial for effectively monitoring and controlling the spread of infectious diseases. In addressing this critical need, our study introduces an engineered application designed to expedite the analysis of bacterial sequencing data, thus providing a streamlined method for species identification. The Bacterial Identifier underwent thorough testing using a significant dataset obtained from the Veterinary Research Institute. Central to the app's functionality is the integration of three essential tools. Implemented through a cohesive bash script, these tools are seamlessly combined to ensure optimal performance and accuracy in bacterial identification. Furthermore, to enhance user experience, a user-friendly interface was developed using Python 3, facilitating intuitive navigation and efficient utilization of the application's capabilities.**

Keywords — Microbiota, Genetic-level identification, Bioinformatic analysis, Bacterial identification app

I. Introduction

Identifying bacterial strains remains an ongoing challenge for human health, especially those carrying antibiotic resistance genes or displaying heightened virulence [1]. Consequently, there is a growing need for genetic-level identification of bacterial strains [2][3].

Besides traditional laboratory techniques used for bacterial identification, bacterial identification can be performed using tools that compare bacterial genomes directly with a reference database. Among these tools, BLAST (Basic Local Alignment Search Tool) is widely used in bioinformatics, especially for smaller databases. However, for larger databases, BLAST [4] may not be suitable due to its long running time. For example, in the study by Kent [5], it is mentioned that to analyze genomes efficiently, rapid mRNA/DNA and cross-species protein alignments are essential. Consequently, the study by Kent [5] introduces a new tool, BLAT, which surpasses popular existing tools such as BLAST in both accuracy and speed, being 500 times faster [5].

This study focuses on the direct development of a software application featuring a user-friendly interface integrated with the bacterial database of the Veterinary Research Institute. The application utilizes BLAST, a traditional and widely used tool in the field, in conjunction with faster tools offering greater heuristic freedom.

Despite its limitations, BLAST remains important in veterinary medicine and future drug research, which is why it is an integral part of our designed application. Additionally, our software app is tailored for rapid bacterial identification, particularly for large databases, and it utilizes the Cd-hit tool [6], which has demonstrated and discussed suitability in our previous research [7][8].

II. Datasets

The robustness of the bacterial database, for which the software application is designed, is broad, as confirmed by a brief summary analysis of the tested dataset. This bacterial database was initially published in the study by Medvedcky et al [9] and expanded with additional pigs samples in the study by Schwarzerova et al [3]. The summary analysis of dataset was constructed using the Up-to-date Bacterial Core Genome (UBCG) [9] by phylogenetic methods and design by IToL v6 [11], see Fig. 1. It represents 452 genome sequences of 8 bacterial phyla – *Firmicutes*, *Bacteroides*, *Verrumicrobiota*, *Actinobacteria*, *Elusimicrobiota*, *Proteobacteria*, *Synergistes* and *Fusobacteria*, providing a comprehensive perspective on the microbiome.

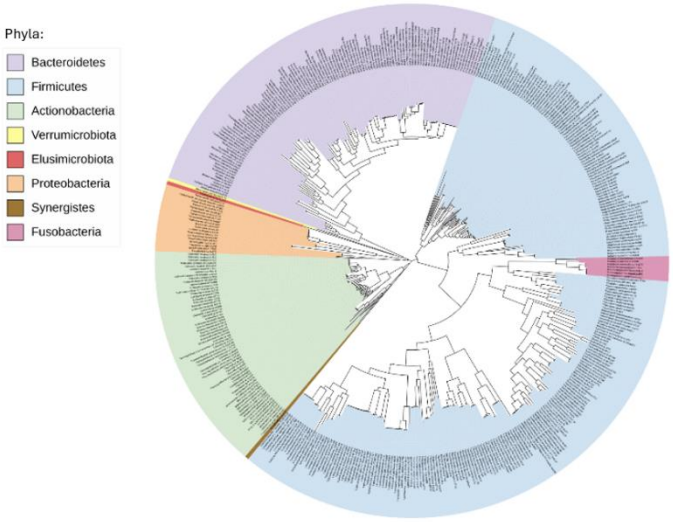

Fig. 1. Phylogenetic tree of 452 sequenced genomes obtained from chicken caecum and pig feces. Bacteria from the phylum Firmicutes (254 genomes) are shown in blue, bacteria from the phylum Bacteroidetes (113 genomes) are shown in purple, the green color represents bacteria from the Actinobacteria phylum (65 genomes), the yellow color represents the Verrumicrobiota (1 genome), the red color represents the Elusimicrobiota (1 genome), the orange color represents bacteria from the Proteobacteria strain (19 genomes), the brown color represents the Synergistes (1 genome) and the pink color represents the Fusobacteria (7 genomes).

III. METHODS

The Bacterial Identifier, a core stone in this study, has been implemented in the bash scripting language. This tool utilizes a combination of specialized software tools, specifically Barrnap [12], Cd-hit [6], and BLAST [4][13]. It was implemented through a cohesive bash script.

The backend of the Bacterial Identifier is structured into multiple phases. In the initial phase, bacterial identification is conducted using specialized tools. Subsequently, in the second phase, a detailed analysis of the output files from Barrnap, Cd-hit, and BLAST tools takes place, enabling a comprehensive understanding of bacterial genomes. In the final phase, a comparative analysis is executed, providing users with definitive identification results.

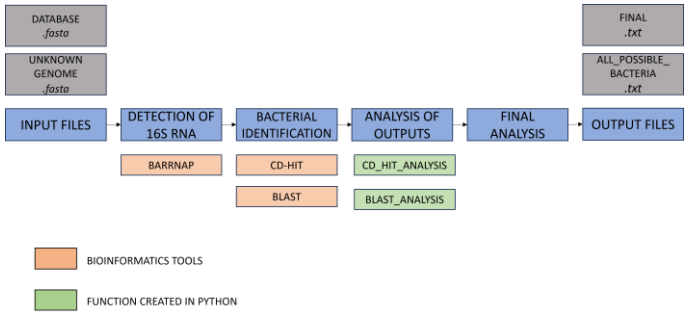

Fig. 2. Block diagram of the Bacterial Identifier. Blue color represents the process of the algorithm, the gray color represents the input and output files, the green color represents functions created in Python 3, and the orange color represents the previous implemented bioinformatics tool.

1) Default Settings of the Bacterial Identifier

To ensure optimal performance and quality results from the Bacterial Identifier, configuring parameters for individual tools such as Barrnap, Cd-hit, and BLAST is crucial. Tables I, II, and III provide a detailed overview of all configured parameters, their functions, and default settings within the Bacterial Identifier.

TABLE I: Barrnap Parameter Settings

Parameter name	Description	Default Setting
kingdom	Kingdom of organisms to be analyzed	bac (bacteria)
evalue	E-value threshold	1e-6
lencutoff	Ratio of minimum predicted rRNA length to standard rRNA length	0.8 (80%)
reject	Threshold for rejecting rRNA (matches with rRNA below this value will be excluded)	0.25 (25%)
outseq	Name of the output file where the predicted sequences will be saved	rRNA.fasta
threads	Number of CPU threads used	6

TABLE II: Cd-hit Parameter Settings

Parameter name	Description	Default Setting
I	Input file (unknown genome)	rRNA.fasta
i2	Input multifasta file with reference genomes	multifasta.fasta
O	Output file	Cd-hit_output
T	Number of CPU threads used will be excluded)	6
M	Maximum memory used by Cd-hit (MB)	0 (unlimited)
C	Sequence similarity threshold	0.95 (95%)

TABLE III: BLAST Parameter Settings

Parameter name	Description	Default Setting
evalue	Minimum match for similarity	1e-5
query	File with unknown bacterium sequences	„$input_file“
db	File with reference bacterium sequences	blast_database
out	Output file	blast.txt
outfm	Output file format	6 qseqid sseqid pident length evalue bitscore
num_threads	Number of CPU threads used will be excluded)	6

2) Output File Analysis

After executing Barrnap, Cd-hit, and BLAST, an analysis of the output files is performed. Due to the different formats of the output files from each tool, two separate Python 3 functions are developed for analyzing Cd-hit and BLAST

outputs. No specific function is created for Barrnap; instead, it is checked whether the output file is empty, ensuring the presence of known rRNA genes in the unknown genome.

Despite the creation of two separate functions – *cd_hit_analysis* and *blast_analysis* – the underlying principle of both functions remains consistent. These functions parse the respective output files, saving only the names of bacteria exhibiting similarity with the unknown bacterium above a predefined threshold. These newly generated files then proceed to the final analysis.

3) *Final analysis*

The final steps of the process are conducted using the core backend of the application in a bash script and vary based on whether the user opts to identify the unknown bacterium using one or more tools after initiating the Bacterial Identifier. If only one tool is selected, the output files of the Bacterial Identifier consist of the results from the functions *cd_hit_analysis* and *blast_analysis*. In the case of multiple tools being chosen, the Bacterial Identifier generates two new output files.

The first file - final.txt, stores bacteria predicted by all selected tools, while the second file - all_possible_bacteria.txt, contains all bacteria predicted by at least one of the chosen tools.

4) *Frontend App Implementation*

To make the Bacterial Identifier accessible to a wider scope of users, a simple and intuitive user interface has been created using the Tkinter library [14] in Python 3. Upon launching the application, the main window appears, serving as a platform to configure parameters influencing the analysis, see Fig. *3*.

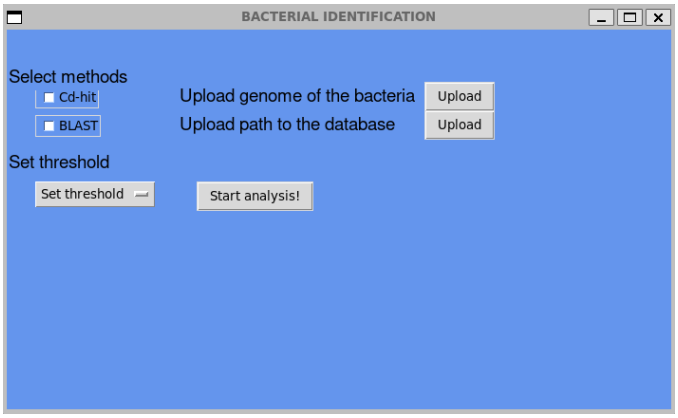

Fig. 3. The window that appears to the user when the application is launched.

In the Selection methods section, users have to the option to choose a method form bacterial identification, either the Cd-hit tool, the BLAST tool, or both simultaneously. If both tools are selected, both are automatically initiated for bacterial identification. Simultaneously, the Barrnap tool is automatically launched to detect 16S rRNA.

Another adjustable parameter is the similarity threshold (*Set threshold*), which determines how similar a sequence must be to be considered the same species. This parameter is optional; if the user does not enter a value, the application automatically sets the similarity threshold to 95%.

On the right side of the app, there are two buttons allowing users to set path to analyzed files. The first button facilitates uploading files with unknown bacteria in *.fasta* format, while the second button enables users to specify the path to reference genomes.

If all parameters are properly configured, users can press the *Start* analysis button, initiating the analysis itself. If the user fails to select a bacterial identification method or set file paths, an error message is displayed, alerting the user to missing configuration, Fig. *4*.

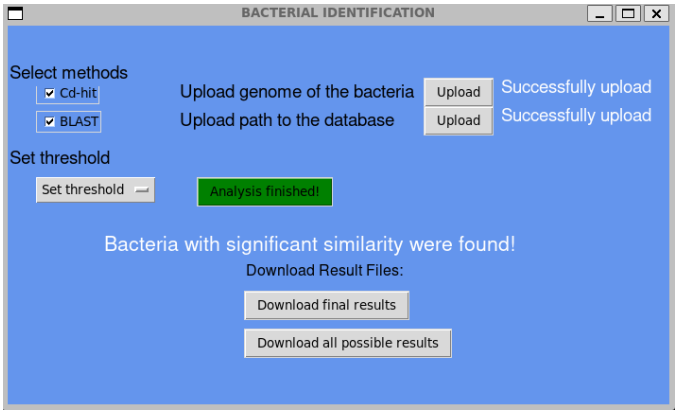

Fig. 4. Example of the message that is displayed if the user does not specify the path to the file with the unknown bacteria and the reference files.

When the user initiates the analysis and it completes successfully, an information message is displayed on the screen, providing the user with a preliminary result of the analysis. This message informs the user whether a match with any of the reference bacteria was found or if the unknown bacteria does not match any of them. Additionally, buttons are displayed that allow the user to download the resulting analysis files, which are crucial for further processing and interpretation of the results, see Fig. *4*.

IV. Results and Discussion

1) *Experiment 1*

To test our app, we randomly selected a bacterium from our analyzed dataset, *Absiella dilichum* ET39, for identification purposes. After running the app, *Absiella dilichum* ET39 was identified as a bacterium from the species *Firmicutes*. Therefore, both Cd-hit and BLAST correctly identified this bacterium.

2) *Experiment 2*

In the second experiment, we uploaded *Bacteroides gallinaceum_25* as an unknown bacterium that was not in the analyzed dataset. *Bacteroides gallinaceum_121* was identified

as the most similar bacterium by the Bacterial Identifier because significant matches with this bacterium were found by both BLAST and Cd-hit.

Cd-hit identified only this bacterium as similar, whereas BLAST identified another bacterium as similar. Specifically, BLAST identified *Bacteroides gallinaceum_121*, *Bacteroides gallinaceum_84*, *Bacteroides gallinaceum_An558*, *Bacteroides gallinaceum_An825*, and *Bacteroides gallinaceum_ET474* as similar bacteria to *Bacteroides gallinaceum_25*.

3) *Comparative Analysis of Computational Efficiency: Cd-hit vs. BLAST in Bacterial Identification*

During the development of the app, a temporal analysis was conducted. According to available sources [15][16], the asymptotic complexity of BLAST is *O(n^2)*, where *n* represents the number of sequences. As the number of sequences increases, there is a substantial increase in the running time of BLAST. Conversely, the Cd-hit tool is expected to deliver faster results when working with larger databases compared to BLAST.

The testing of the Bacterial Identifier was performed on the Veterinary Research Institute database, comprising 452 bacterial genomes. For the linear model, the R2 value was 0.99626 for Cd-hit and 0.94138 for BLAST. However, when we calculated the R2 values for the exponential model, the R2 value for Cd-hit was only 0.76779, whereas for BLAST it was 0.94406. From these calculated R2 values, it is evident that the runtime of Cd-hit increases linearly with the database size, whereas for BLAST, it increases exponentially. This insight suggests that when working with a more extensive test dataset, Cd-hit would represent a faster alternative to BLAST.

V. Conclusion

In conclusion, this study has presented the development and evaluation of the Bacterial Identifier application, designed to streamline bacterial identification processes. By integrating tools such as BLAST and Cd-hit and incorporating a user-friendly interface, the Bacterial Identifier offers an efficient solution for researchers and practitioners in the field of microbiology. Through testing on the Veterinary Research Institute database, consisting of 452 bacterial genomes, we have demonstrated the effectiveness of the Bacterial Identifier in accurately identifying bacterial strains. Our analysis revealed insights into the computational performance of BLAST and Cd-hit, with Cd-hit exhibiting linear scalability and potentially faster runtime when handling larger datasets compared to the exponential increase observed with BLAST.

The results underscore the importance of considering computational efficiency when selecting tools for bacterial identification, particularly in scenarios involving extensive datasets. The Bacterial Identifier, with its emphasis on speed, accuracy and user-friendly enviroment, presents a valuable resource for researchers and practitioners seeking to expedite the identification process while maintaining high-quality results.

Acknowledgment

Computational resources were provided by the e-INFRA CZ project (ID:90140), supported by the Ministry of Education, Youth and Sports of the Czech Republic

This work has been supported by the National Centre for Biotechnology in Veterinary Medicine – NaCeBiVet.

References

[1] TERRENI, Marco; TACCANI, Marina; PREGNOLATO, Massimo. New antibiotics for multidrug-resistant bacterial strains: latest research developments and future perspectives. *Molecules*, 2021, 26.9: 2671.

[2] LI, Wenjun; RAOULT, Didier; FOURNIER, Pierre-Edouard. Bacterial strain typing in the genomic era. *FEMS microbiology reviews*, 2009, 33.5: 892-916.

[3] SCHWARZEROVA, Jana, et al. Detecting horizontal gene transfer among microbiota: an innovative pipeline for identifying co-shared genes within the mobilome through advanced comparative analysis. *Microbiology Spectrum*, 2024, 12.1: e01964-23.

[4] ALTSCHUL, Stephen F., et al. Basic local alignment search tool. *Journal of molecular biology*, 1990, 215.3: 403-410.

[5] Kent, W.J., 2002. BLAT—the BLAST-like alignment tool. *Genome research*, *12*(4), pp.656-664.

[6] Li, W. and Godzik, A., 2006. Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics*, *22*(13), pp.1658-1659.

[7] SCHWARZEROVÁ, Jana, et al. Systems biology approach for analysis of mobile genetic elements in chicken gut microbiome. In: *2022 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. IEEE, 2022. p. 2865-2870.

[8] SCHWARZEROVÁ, Jana, et al. A minireview on the bioinformatics analysis of mobile gene elements in microbiome research. *Frontiers in Bacteriology*, 2023, 2: 1275910.

[9] MEDVECKY, Matej, et al. Whole genome sequencing and function prediction of 133 gut anaerobes isolated from chicken caecum in pure cultures. *BMC genomics*, 2018, 19.1: 1-15.

[10] Kim, J., Na, S.I., Kim, D. and Chun, J., 2021. UBCG2: Up-to-date bacterial core genes and pipeline for phylogenomic analysis. *Journal of Microbiology*, *59*(6), pp.609-615.

[11] LETUNIC, Ivica; BORK, Peer. Interactive Tree Of Life (iTOL): an online tool for phylogenetic tree display and annotation. *Bioinformatics*, 2007, 23.1: 127-128.

[12] SEEMANN, T.; BOOTH, T. Barrnap: basic rapid ribosomal RNA predictor. *GitHub repository*, 2018.

[13] Madden, T., 2013. The BLAST sequence analysis tool. *The NCBI handbook*, *2*(5), pp.425-436.

[14] LUNDH, Fredrik. An introduction to tkinter. *URL: www. pythonware. com/library/tkinter/introduction/index. htm*, 1999.

[15] CHAN, Alexander. An Analysis of Pairwise Sequence Alignment Algorithm Complexities: Needleman-Wunsch, Smith-Waterman, FASTA, BLAST and Gapped BLAST. URL: https://biochem218.stanford.edu/Projects%202004/Chan.pdf.

[16] *What is the time complexity of the given BLAST?* Online. In: Stack overflow. URL: https://stackoverflow.com/questions/59454358/what-is-the-time-complexity-of-the-given-blast.