BMC Bioinformatics

RESEARCH Open Access

Cascade PSI-BLAST 2.0: a fast-searching parallelized remote homology detection tool and development of Cascade web server 2.0

Soumya Nayak[1], Dheemanth Reddy Regati[1], Murugavel Pavalam[1] and R. Sowdhamini[1,2,3*]

*Correspondence:
R. Sowdhamini
mini@ncbs.res.in
[1]National Centre for Biological Sciences (Tata Institute of Fundamental Research), GKVK Campus, Bangalore, Karnataka 560065, India
[2]Molecular Biophysics Unit, Indian Institute of Science, Bangalore 560012, India
[3]Institute of Bioinformatics and Applied Biotechnology, Bangalore, India

Abstract

Background Remote homology detection is critical for inferring evolutionary and functional relationships among proteins, but divergence below 30% identity often hinders standard sequence-based searches. Profile-based methods and HMMs improve sensitivity, yet many distant homologues remain undetected. Cascade PSI-BLAST uses efficient, iterative, multi-generation searches of intermediate hits to bridge sequence gaps, constructing successive PSSMs that reveal remote relationships without structural information. However, its application to ultra-large databases such as NR and UniProt is computationally intensive. We address this limitation by optimizing intermediate selection and enabling distributed execution across multiple CPUs or servers, significantly enhancing throughput and resource utilization while preserving detection sensitivity.

Results We applied the parallelized Cascade PSI-BLAST algorithm to sequences drawn from multiple SCOP classes to evaluate its sensitivity and predictive power on the GenDis database. Our method uncovers substantially more remote homologues, with less false positives, in both GenDis and UniProt compared to conventional searches. Large repositories such as NR can also be processed in a practical timeframe by distributing cascade searches across multiple servers. Finally, we have deployed a user-friendly web server for running cascade searches on smaller databases, including PDB and Swiss-Prot.

Conclusion Our enhanced Cascade PSI-BLAST markedly improves detection of remote homologues across both large and curated databases by leveraging intermediate sequences and distributed execution. Multi-server parallelization reduces runtimes on expansive repositories such as NR to practical levels. The web server offers rapid, user-friendly searches on smaller datasets, while the standalone package supports scalable, customizable analyses on local infrastructure. Together, these tools provide a versatile, sequence-only platform for uncovering distant protein relationships, accelerating functional annotation and evolutionary insights.

Keywords PSI BLAST, Cascade PSI_BLAST, Parallel processing, GNU parallel, Remote homology search, Webserver

© The Author(s) 2026. **Open Access** This article is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License, which permits any non-commercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if you modified the licensed material. You do not have permission under this licence to share adapted material derived from this article or parts of it. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by-nc-nd/4.0/.

Background

Homologous proteins are related to each other at sequence, structure, and functional level. Proteins that are divergently evolved and related at the superfamily level often display low sequence identity, but high similarity in structural topology and function. Due to high evolutionary divergence and low sequence identity, it is often challenging to obtain reliable sequence alignments for distantly related proteins.

If the sequence identity shared by proteins is very low (< 30%), it is difficult for sequence-based search methods to identify such proteins thoroughly. To detect these relationships, various approaches have been employed such as position-specific profiles [1, 2], Hidden Markov models (HMMs) [3], and intermediate sequences [4]. A remote homology search approach Cascade PSI-BLAST [5, 6] that identifies such relationships has been previously developed in our lab. This establishes the relationships using a cascaded search and is entirely a sequence-based method. This approach rigorously implements an intermediate sequence-based search procedure through the extensive application of PSI-BLAST [7]. In these procedures, intermediate sequences can bridge the sequence gap between distantly related proteins.

PSI-BLAST (Position-Specific Iterated BLAST) is an extension of the BLAST (Basic Local Alignment Search Tool) algorithm. It is particularly useful for detecting distant evolutionary relationships between proteins by using position-specific scoring matrices (PSSMs) [8]. These PSSMs are then used to identify hits in subsequent iterations.

In the Cascade PSI-BLAST search method, a typical PSI-BLAST search, initiated with a query, is followed by multiple PSI-BLAST runs. The follow-up runs would be structured such that each of the hits of the first "generation" becomes individual queries. Here, one generation refers to one PSI-BLAST run with its own iterations. In the next generations, there will be further PSI-BLAST runs, where new hits are provided a chance to become queries leading to more hits until there are no more new queries. In each generation, intermediate sequences are found and used to continue the searches that identify distant hits. Since every intermediate sequence is given a chance to scan the protein sequence space, this search method is highly effective at making connections within protein superfamilies and folds.

The performance of Cascade PSI-BLAST on large databases like NR [9] and UniProt [10], was challenging with the previous versions of cascade sequence searches [5, 6], giving rise to a dire need for improved platforms of support systems. In the paper, we have implemented various strategies such as selecting optimum intermediate hits for downstream generation and utilising parallel processing across multiple CPUs and even connecting multiple servers. This allows queries in each generation to run simultaneously on different servers. This approach is crucial because each generation typically yields hundreds of hits, and each hit requires a significant amount of time to process within a large database. The Cascade web server has been updated with the latest databases. The new algorithm implemented in the Cascade PSI-BLAST web server enhances search speed. The web server can be accessed from https://caps.ncbs.res.in/cascadePsiBlast

Original Cascade PSI-BLAST algorithm

As explained in the previous section, the Cascade PSI-BLAST algorithm operates by continuously propagating PSI-BLAST searches through hits identified at the end of each generation until no new hits are detected (Fig. 1). Initially, a single query is used, and

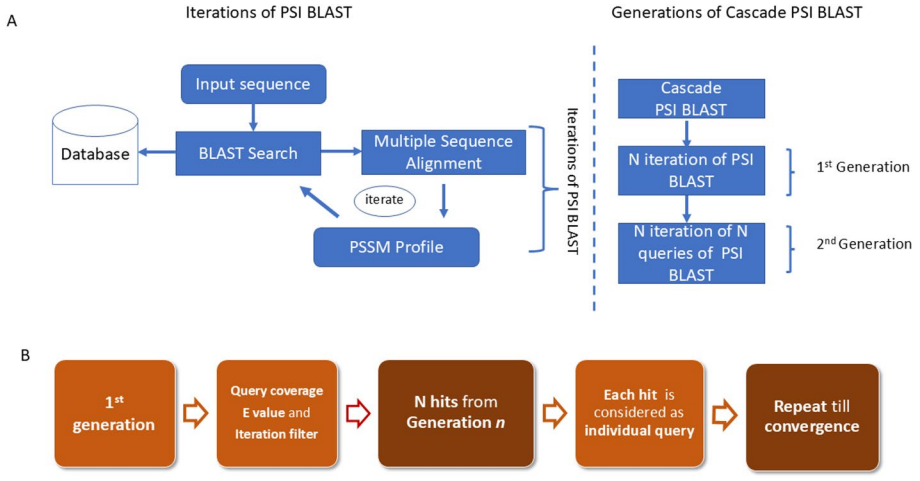

Fig. 1 The original Cascade PSI-BLAST Algorithm. **A** shows the difference between the Iterations of PSI-BLAST and the generations of Cascade PSI-BLAST. IN PSI-BLAST a PSSM profile is created from the multiple sequence alignment of the hits generated from the query and this PSSM is used in the subsequent search till the search converges. Each cycle is an iteration. Cascade PSI-BLAST uses the result of N—iterations as one generation. **B** shows the CASCADE PSI-BLAST algorithm originally used

after N iterations of PSI-BLAST runs, the hits identified are considered as the results of the first generation search. All hits from this initial search become queries in the 'second generation'. Before proceeding to the next generation, query coverage (typically 75%) and an E-value filter (typically 0.01) are applied to all hits to ensure false positives are minimized. The propagation of PSI-BLAST for a single query can result in 500 to 1000 individual PSI-BLAST searches, through all detected hits over multiple generations, depending on the databases used for the search [5, 6].

Methods

Improvements and new feature added

We have implemented the following features as improvements or support to the existing software to enable the runs to go faster and also on larger sequence databases.

The algorithm is modified and rewritten in Python so that the program becomes platform-independent. The added features are as follows:

- CD hit module [11] is incorporated so that highly similar sequences do not enter into independent PSI-BLAST runs since they are likely to generate redundant results. This also helps to reduce the number of runs in a generation. The default cut-off is 0.7 but the user can use any cut-off from 0.65 to 0.95.
- The program can now restart from any generation
- The user can give any number of hits to propagate to the next generation depending on the CPU power
- We have also used parallel processing in one or multiple servers to speed up the search runs

Algorithm

The algorithm is initiated using one single query (please see Fig. 2). After the first generation, all the hits were filtered using the query coverage and E-value filter. Query coverage is the percentage of the query sequence that is aligned with the subject sequence(s)

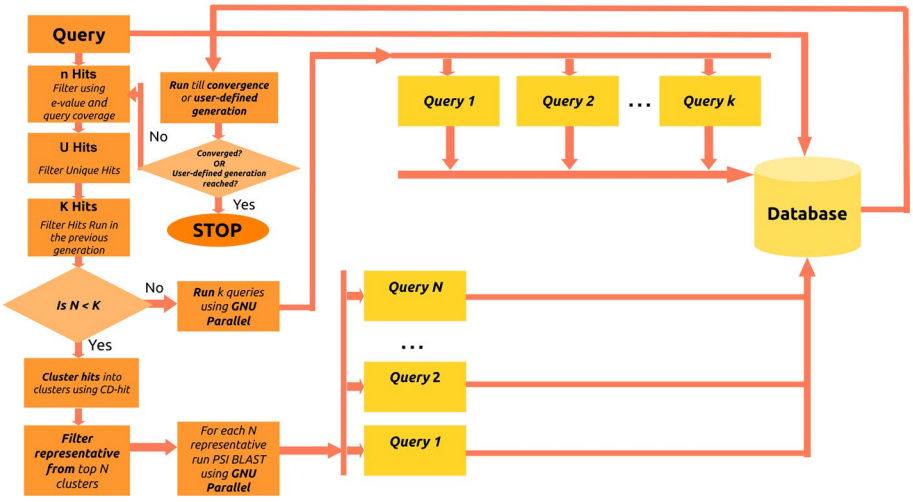

Fig. 2 The updated Cascade PSI-BLAST algorithm

Table 1 List of parameters and their thresholds

Parameters	Default value and accepted range
E value	0.01 (Any value less than 1)
Query Coverage	75% (60–90%)
Cd-hit clustering percentage	70 (65–95)
Database	Any database of protein sequences in BLAST format
Subsequent hits to propagate for next generation	N (the default is 50). Can vary depending on system capacity and user requirements

in the database. These filters are added to minimize the false positives in the subsequent runs. This is important as propagating any smaller or statistically insignificant hit can result in accumulating false positives. The default values for query coverage and E value is 75% and 0.01 respectively but can vary depending on the user choice. A list of parameters and thresholds is given in Table 1. The number of hits to propagate to the next generation is decided by the user. In each generation, the queries were further filtered to ensure that this was not run in any previous generations. "K" is defined as the hits obtained after applying all the filters for each generation. If the number of queries to run in the next generation ("N" which is defined by the user) is less than K value (the number of hits after filtering in each generation), then the CD-HIT module is run to cluster the queries. For example, if N = 50 and K = 300, then the user wants 50 hits to propagate for the next generation, but there are 300 hits. In this case, all 300 hits were clustered, the clusters were arranged according to size and representatives from the top 50 clusters were selected as queries for the next generation. All 50 queries were run in parallel on multiple servers. In cases, N > K, the user wants more hits to propagate than the filtered hits available to consider for the next generation. In this case, all the K hits available are run in parallel (Fig. 2). If the database is large (like NR or UniProt), then the hits obtained after each generation(K) are large. It is not feasible to run all the queries. The hits are clustered and N representative (user-defined) hits from the clusters are selected.

All the hits can be run in parallel in multiple servers, depending on the availability of resources for a user. The complete user manual is provided in Supplementary File 1 and instructions for running the webserver is in help section of the webserver. The

interpretation of results is provided in both the sections. The web server can accept runs on smaller databases like SwissProt [12], SCOP [13] and PDB [14]. The results can be then downloaded from the web server. For larger databases like UniProt and NR, the user can download the executables from the download button of the web server and can run on their own servers and machines.

Parallelization of the Cascade PSI-BLAST software

Sequence databases like NR and UniProt are extremely large, often exceeding hundreds of gigabytes. Running CPU-intensive processes, such as Cascade PSI-BLAST on these databases, requires substantial server resources. Even with access to powerful servers, equipped with multiple CPUs, optimal utilization of computational power necessitates the implementation of parallel processing techniques in the coding.

The processor is a critical component of any computer system responsible for executing instructions and managing tasks. Processing can be categorized into two main types: serial and parallel.

Serial processing involves the processor completing tasks one at a time in a sequential manner. Each task waits for the previous one to finish before starting, leading to a linear execution of tasks. This method is known as serial processing or sequential processing.

In contrast, by definition, in parallel processing, multiple processors execute tasks simultaneously. Each processor works independently on its assigned tasks, allowing for concurrent processing. This approach significantly enhances throughput and reliability, since failures in one processor do not affect others. Most modern computers are equipped with multiple CPUs, enabling the implementation of parallel processing techniques. By default, Python code typically runs on a single processor. To fully exploit the available computing power, multiprocessing techniques can be employed, utilizing threading or multiple processors to execute independent parts of the program concurrently.

A thread is a unit of execution within a process that shares the process's memory space with other threads. PSI-BLAST already employs threading internally, where each thread processes a subset of the database. However, certain phases of the process, such as setup and formatting, may still operate on a single thread. Despite the use of multithreading, we have observed that a significant portion (20–30%) of CPU resources remain underutilized. Therefore, adopting multiprocessing programming allows us to harness the full capabilities of available CPUs and accelerate processing speeds effectively.

Parallelizing using GNU parallel

GNU Parallel [15] is a command-line tool designed for executing tasks concurrently across one or more computers. Each task can consist of a single command or a small script that needs to be run for each item in a list of inputs. Multiple tasks can run simultaneously in parallel.

Using GNU Parallel enhances memory and CPU management efficiency. For example, if there are 32 tasks to execute and 4 CPUs available, straightforward parallelization might lead to CPU idle time depending on task execution durations. Instead, GNU Parallel dynamically spawns new processes as tasks complete, ensuring that CPUs remain active and thereby optimizing overall processing time.

We have currently been running the software upto three servers, but the users can connect to many more. The source code can be downloaded from the webserver and used to connect to several servers. When connecting to multiple servers, several key considerations must be addressed:

- *Synchronization*: Ensuring synchronization between servers is crucial because we are running a single codebase across multiple servers, each handling independent units. The return values from each server need to be carefully monitored to ensure there are no delays and that all processes are completed successfully.
- *Thread vs. Process creation optimization*: It is essential to create an optimal number of threads and processes. This optimization ensures that managing coordination between them does not consume excessive time, which could delay execution.
- *State management*: Effective state management of all threads and processes is necessary to prevent errors and exceptions.

All these issues were carefully addressed to ensure smooth and error-free execution when connecting and coordinating multiple servers.

Validation and comparisons

In order to compare Cascade PSI-BLAST with other methods using statistical measures and present for case studies, we have employed the results available in GenDis + [18] as the ground truth. However, GenDis + is a pre-curated resource that has gathered homologues using multiple sequence search methods, such as CS-BLAST and PHI-BLAST, followed by detailed computational validations. Statistical parameters such as coverage and realisation of false positives, in comparison with the ground truth, is provided for comparison with other methods.

Results

Performance gain by parallelizing the PSI-BLAST

In the above table (Table 2), the performance improvement after parallelizing PSI-BLAST is discussed. Since PSI-BLAST has its own threading capabilities, we measured the performance improvement after utilizing the maximum number of threads to avoid overloading the server. Table 2 compares the running time using a small database (GenDis) versus a large database (UniRef90). It also provides a comparison of running times between a small machine and a large server.

For each run, the query file consists of a FASTA file with 30 sequences, simulating a generation run for cascade PSI-BLAST using 30 queries for propagation to the next generation. The specifications of the machines are as follows:

- Server Specifications

Processors: 56.
RAM: 80 GB

- Small machine Specifications

Processors: 8
RAM: 16 GB

Table 2 Comparisons of 30 PSI-BLAST run in small/large servers and small versus large databases

Database	Small machine (only threading): No parallel process, 7 threads	Small machine (only parallel process): 4 parallel processes, no threading	Small machine (parallel process + threading): 3 parallel processes, 3 threads	Server (only threading): 54 threads	Server (only parallel process): 50 processes	Server (parallel process + threading): 30 processes + 30 threads
Small (GenDis, 17 GB)	63 min 27 s	No gain	58 min 57 s	11 min 21 s	No gain	9 min 36 s
Large (Uniprot90, 82 GB)		No gain		67 min 40 s	No gain	61 min 22 s

It is observed that maximum gain in time is observed while using threading and parallel processing. This is particularly true for running queries on a large database on a large server (67 min using only PSI-BLAST inherent threading vs. 61 min using threading and parallel processing). However, it is also true that the gain obtained is not significant and the only solution to truly speed up the process for running against large databases is to go for execution of the process in multiple servers simultaneously. It is also observed that the process is not memory-bound at any point in time and only 5%-8% of memory is occupied.

Coverage analysis of PSI-BLAST versus Cascade PSI-BLAST

To analyze the coverage gained by cascade PSI-BLAST over PSI-BLAST, we have chosen a few superfamilies from the GenDiS + [18] database for testing (Supplementary File 2). GenDiS (Genomic Distribution of Protein Structural Domain Superfamilies) database provides a projection of SCOP superfamily members onto the sequence space of the NR database from NCBI. In GenDis, the sequences in the NR database are organized into SCOP superfamilies. These sequences are accumulated for each superfamily using various methods such as PHI BLAST and CS BLAST for each superfamily using multi-query approach. For this study, we have selected eight superfamilies from SCOP, representing four different protein classes. The protein classes are defined as follows:

1. *All alpha class*: This contains protein domains in which the secondary structure is predominantly alpha helices.
2. *All beta class*: These protein domains are primarily composed of beta-sheets.
3. *Alpha and beta*: These include protein domains that contain both alpha helices and beta strands. These proteins are characterized by the combination of these two secondary structural elements.
4. *Alpha/beta*: protein domains that contain both alpha helices and beta strands arranged in repeating structural motifs. These proteins typically have alternating alpha helices and beta strands.

In Table 3, the coverage of superfamilies 51,101 (Mannose-binding Lectins) and 51,351 (Triosephosphate Isomerase) is presented. PSI-BLAST and Cascade PSI-BLAST were run for individual domains, and the coverage per domain was provided. The cumulative data represents the total coverage from all domains after removing repetitions and false positives, if any. It is evident that, for each superfamily and all domains, Cascade PSI-BLAST achieves over 50% more coverage than PSI-BLAST alone. The results of the cascade PSI BLAST run is provided in https://caps.ncbs.res.in/download/cascade-parallel/supplementary_file_3/

Figure 3 presents case studies from the above two superfamilies, where the hits identified by cascade PSI-BLAST have very low identity but similar structure and function annotation. These hits cannot be identified directly from PSI-BLAST but can be identified by the intermediate sequence search approach of cascade PSI-BLAST.

A representative hit for superfamily 51,101 (Mannose-binding Lectins) is (NP_001185202.1 Mannose-binding lectin superfamily protein [*Arabidopsis thaliana*]) (Fig. 3A). This superfamily is from all beta SCOP classes. The representative query d1c3ma_ from this superfamily and the identified hit have low sequence identity (~ 28%).

Table 3 PSI-BLAST versus Cascade PSI-BLAST comparison for all domains of Two superfamilies 51,101 (Mannose-binding Lectins) and 51,351 (Triosephosphate Isomerase)

Superfamily 51101
No. of sequences: 3121
No. of members: 4

Members	PSI_BLAST	% coverage	Cascade PSI-BLAST	% coverage
d1c3ma	679	21.76	1990	63.7
d1ouwa	682	21.85	1604	51.3
d1ugx1	615	19.7	1615	51.74
d2guda1	781	25.02	2344	75.1
Cumulative	1248	39.9	2547	81.6

Superfamily 51351
No. of sequences: 7600
No. of members: 3

Members	PSI_BLAST	% coverage	Cascade PSI-BLAST	% coverage
d1w0ma	553	07.28	4496	59.1
d1aw1a	672	8.84	1380	18.1
d1n55a	1456	19.1	1798	23.6
cumulative	**1949**	**25.6**	**5862**	**77.1**

Fig. 3 The secondary structure annotation of the queries (d1c3ma_ and d1woma_) and representative hits from two superfamilies 51,101 (Mannose-binding lectins) and 51,351 (Triosephosphate Isomerase) which belongs to all beta and alpha/beta SCOP class respectively. The two identified hits have low sequence identity (~ 28%) with the query and cannot be identified directly by PSI BLAST. But these are identified using the intermediate sequence search method of Cascade PSI BLAST

Despite low sequence identity, they have similar functional and structure annotations (Fig. 3A).

Another example is from superfamily 51,351 (Triosephosphate Isomerase) which belongs to SCOP class alpha/beta. The secondary structure annotation of representative hit WP_032984714.1 and the query d1w0ma_ is shown in Fig. 3B. The sequence identity is low (26%) but both the query and the identified hit have a similar structure and functionally both are annotated as Triosephosphate Isomerase. In both cases, simple PSI-BLAST failed to recognise these connections at similar E-values.

Figure 4 illustrates the data for eight superfamilies belonging to different SCOP classes. It is observed that for the all-alpha, all-beta, and alpha/beta classes, Cascade PSI-BLAST achieves approximately 73% coverage on average, whereas PSI-BLAST achieves around 35% coverage. For the class 'alpha and beta', although both PSI-BLAST and Cascade

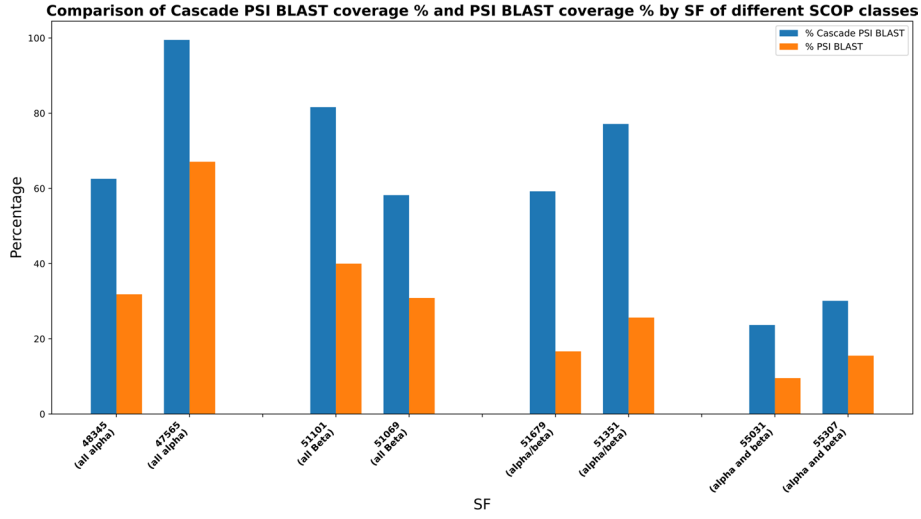

Fig. 4 Comparison of coverage percentage of PSI-BLAST and Cascade PSI-BLAST by superfamily of different SCOP classes

PSI-BLAST show lower coverage, Cascade PSI-BLAST still identifies more hits (Cascade PSI-BLAST results can be viewed in Supplementary File 4).

Performance analysis of Cascade PSI-BLAST for a multidomain protein in a large database (UniProt)

We have assessed the scalability of Cascade PSI-BLAST 2.0 when querying a large database in UniProt. UniProt [10] is a comprehensive resource that offers extensive information on protein sequences and their functional annotations. Its core databases include UniProtKB (Knowledgebase), UniRef (UniProt Reference Clusters), and UniParc (UniProt Archive).

UniRef [16], specifically UniRef90, organizes protein sequences from UniProtKB that share 90% or higher sequence identity into clusters. These clusters reduce redundancy by consolidating similar sequences into representative entries, facilitating efficient navigation and analysis of large protein datasets. Utilizing UniRef enhances the speed and effectiveness of database searches and analyses while preserving essential biological information from the original sequences. Hence, we chose to utilize the UniRef database for our sequence searches.

For our test case, we selected Prolyl oligopeptidase (pdb id 2bkl), a serine peptidase belonging to the Superfamily S9 family as classified by MEROPS [17]. This enzyme exhibits a two-domain architecture consisting of a catalytic domain and an N-terminal Beta propeller domain. The entire sequence of Prolyl oligopeptidase serves as our query for evaluation purposes.

Figure 5 represents the results obtained in PSI-BLAST versus Cascade PSI-BLAST. PSI-BLAST was able to identify 819 hits, while Cascade PSI-BLAST running for 5 generations was able to identify 2888 hits (Fig. 5A). It should be noted that the query was run in the UniRef 90 cluster database. The run identifies the representative hits and expanding these clusters can result in an increased number of hits. It is observed that most of the intermediate sequences were identified in generations 2 and 3 (Fig. 5B). The figure shows the cumulative results after completing each generation and the new hits

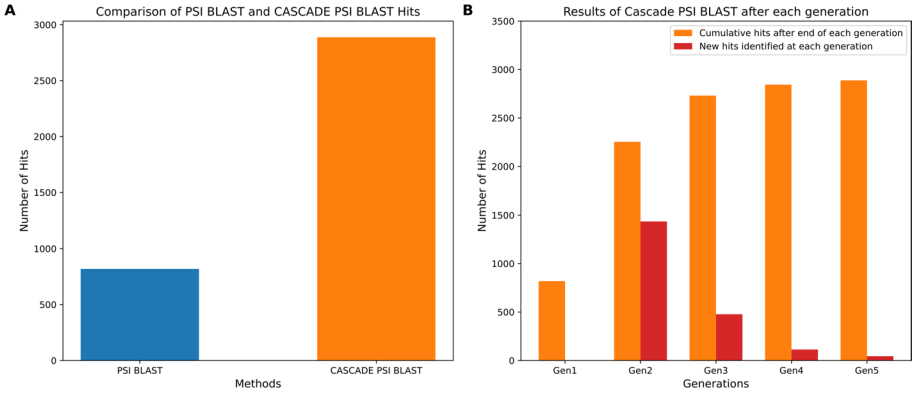

Fig. 5 Comparison of the number of hits in PSI-BLAST and Cascade PSI-BLAST in the database UniRef 90 (**A**). The cumulative hits from Cascade PSI-BLAST after each generation (**B**)

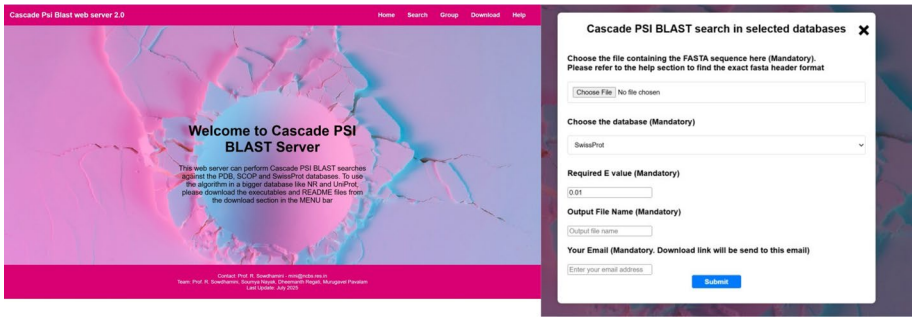

Fig. 6 Snapshot of homepage of Cascade PSI-BLAST webserver 2.0 and the search options with the mandatory and optional parameters

identified in each generation. The new hits identified in each generation are considered intermediate hits which, consequently, are considered as queries to the next generation.

Comparison of Cascade PSI-BLAST to other methods

As shown in Fig. 4, Cascade PSI-BLAST outperforms regular PSI-BLAST for a number of protein domain superfamilies belonging to diverse folds and structural classes. We next compared the performance of Cascade PSI-BLAST with others like mmseqs2 and phmmer (shown in Supplementary File 5). The coverage could be equally good for the three methods, but significantly, the number of false positives are much lower for Cascade PSI-BLAST (high specificity) which reduces the burden of further computational validations. We also present the results when applied on two other superfamilies belonging to different structural classes (please see Supplementary File 6) where the coverage is significantly better than a regular PSI-BLAST run. This shows that this method can be reliably used, in general, for diverse set of protein domains.

The Cascade PSI-BLAST web server

The web server implementation of Cascade PSI-BLAST can be accessed at the following URL:

https://caps.ncbs.res.in/cascadePsiBlast

The online Cascade PSI-BLAST Web Server offers the following features (Fig. 6).

- A text box that accepts user-defined input sequences in FASTA format.

- Options to initiate searches in three databases: SwissProt, SCOP, and PDB. Larger databases such as the non-redundant database (NR) and UniProt, which could compromise speed and time, are not included. Users can download the Cascade PSI-BLAST software to run locally for these larger databases.
- Users can set the E-value for the search. The default E-value is 0.01.
- The server defaults to a 75% query coverage filter to reduce short-length hits that may lead to false positives in subsequent analyses. Users have the option to adjust this filter within the range of 60–90%.
- To manage computing time limitations due to multiple server requests, each search can seed up to 30 sequences at a time, with a maximum of 5 generations per search.
- Once the search is complete, results can be downloaded. Result files remain on the server for two days and a download link is also emailed to the user. Detailed steps for interpreting the results are provided in the help section.
- The options to email the results once the run if over is also provided. The email link remains valid for 3 days after the search is over.
- The help section provides the details for the execution of the server and interpreting the results.

Discussion

We have demonstrated that leveraging intermediate hits in a cascaded search strategy can overcome many of the limitations inherent in standard remote-homologue detection methods. We have augmented the results from our previous studies and shown that connection between the distant homologues can be accurately made without depending on the structural data by leveraging the methods using intermediate hits. This has huge implications, as the sequence database usually grows exponentially in comparison to structural database. This is particularly valuable with the exponential growth of sequence databases compared to their structural counterparts: our approach expands the detectable homologue space far beyond what PSI-BLAST alone can achieve.

Quantitatively, the cascaded pipeline recovers more remote homologues than a single-round PSI-BLAST search, without sacrificing specificity and less false positives. Moreover, we find that even after fully exploiting PSI-BLAST's multi-threading capability, on average 20–30% of CPU resources remain unutilized. This can be harnessed by running multiple PSI-BLAST jobs in parallel, allowing users to optimize for time management.

However, when applying Cascade PSI-BLAST to very large databases such as NR, a single server—even a multicore one—may not deliver results within a practical timeframe. We therefore implemented support for distributed execution via GNU parallel: by connecting multiple servers (or even multiple desktop workstations), users without access to high-end servers can still process large-scale searches efficiently. This capability broadens the reach of large-database homology searches, enabling laboratories with modest computational resources to run them effectively. In addition, cascade PSI-BLAST can be used to annotate the hypothetical proteins in the sequence databases by connecting them to their distant homologues.

Finally, to facilitate adoption and further development, we provide the complete source code. Our framework is modular: replacing the PSI-BLAST module with an HMM-based search (or any other sequence-search tool) requires only a trivial modification.

As new algorithms emerge, they can be slotted into the cascade to continue improving remote-homologue coverage without reengineering the entire workflow.

Conclusion

Proteins can evolve in a way that preserves their function and structure while significantly diverging in terms of sequence. Hence, it is difficult to establish homology between proteins with very low sequence similarity. However, intermediate sequences can act as missing links and can identify more distantly related proteins. With the abundance of genomes available in sequence databases, it is possible to leverage these databases to perform more rigorous sequence searches using intermediate hits. Cascade PSI-BLAST is designed to identify such intermediate sequences and use them to identify distantly related homologous proteins. This can accelerate the functional annotation and evolutionary insights.

We have enhanced the existing Cascade PSI-BLAST algorithm by adding new features and optimizing it to run in a parallelized manner across multiple servers. This allows for scaling searches in large databases. Additionally, a new web server implementation is provided, enabling users to perform quick web searches for smaller databases. Users can also download and run the software on their servers to identify hits in larger databases.

Abbreviations

HMMs	Hidden Markov models
PSI-BLAST	Position-Specific Iterated BLAST
BLAST	Basic Local Alignment Search Tool
PSSMs	Position-specific scoring matrices
NR	Non-redundant database
GenDiS	Genomic Distribution of Protein Structural Domain Superfamilies

Supplementary Information

The online version contains supplementary material available at https://doi.org/10.1186/s12859-026-06366-7.

Supplementary Material 1: General instructions about running the webserver as well as interpreting results from the search. There are also instructions on how to download, run and interpret results of multiple server cascade PSI-BLAST standalone version.

Supplementary Material 2: Information about the SCOP superfamilies used for the study shown in Figure 4.

Supplementary Material 3: Supplementary file 3 contains the result of cascade-PSI BLAST run for table 3 (51101 (Mannose-binding Lectins) and 51351 (Triosephosphate Isomerase).Considering the size of the whole result files, which is in GBs, only the final results which includes the total unique outputs from the run and the generation directories which contains the intermediate hits for the next run are provided in the supplementary file. The whole results folder for each run can be downloaded from the webserver.For a full example, the full output for 51351 is provided.This can be obtained from https://caps.ncbs.res.in/download/cascade-parallel/supplementary_file_3/.

Supplementary Material 4: Supplementary file 4 contains the Cascade PSI-BLAST results for all the runs from the SCOP superfamily in the same manner as described in supplementary file 3. The full run files which includes all the PSI-BLAST hits file can be downloaded from the webserver.This can be obtained from https://caps.ncbs.res.in/download/cascade-parallel/supplementary_file_4/.

Supplementary Material 5: Comparisons between Cascade PSI-BLAST, phmmer and mmseqs2 with similar data as Table 3. We compare total hits, hits mapping to superfamily and number of false positives. Percentage coverage for two domains in two superfamilies (51101 and 51351).

Supplementary Material 6: Results from the Cascade-PSI BLAST runs, in tune with Table 3, for the superfamilies 48345 (virus capsid protein alpha helical domain), 47565 (Insect pheromone/odorant-binding proteins) and 55239 (RuBisCO, small subunit).

Acknowledgements

The authors would like to acknowledge NCBS (TIFR), SERB, IBAB and DBT for infrastructural and other support. We thank Mr Rajesh R for help in the construction of the database front-end. This paper is dedicated to late Prof N. Srinivasan for initiating this project.

Author contributions
RS designed the experiments and conceived the idea. SN, DRR performed the experiments, analysed the data. SN, DRR wrote the draft of the manuscript and RS improved it. Murugavel Pavalam has written the first version of the code.

Funding
This work was supported by PhD fellowship from Department of Biotechnology (DBT). Authors thank NCBS for infrastructural facilities. RS is a J.C. Bose National Fellow (JBR/2021/000006) from the Science and Engineering Research Board, India. RS would also like to thank Bioinformatics Centre Grant funded by the Department of Biotechnology, India (BT/PR40187/BTIS/137/9/2021) and the Institute of Bioinformatics and Applied Biotechnology for the funding through her Mazumdar-Shaw Chair in Computational Biology (IBAB/MSCB/182/2022).

Data availability
Data used for this study and results generated are available in supplementary material and caps/download area (https://caps.ncbs.res.in/download/cascade-parallel/). The links for the datasets used in the study are mentioned below: Gendis+ database [18]: https://caps.ncbs.res.in/cgi-bin/gendis+/help.py?help=gendis_download&help_t=Gendis+%20Dowloads; https://caps.ncbs.res.in/download/cascade-parallel/all_gendis.fasta. UniRef90 [10]: https://ftp.uniprot.org/pub/databases/uniprot/uniref/uniref90/uniref90.fasta.gz. Availability and Requirements: Project name: Cascade PSI-BLAST. Project home page: https://caps.ncbs.res.in/cascadePsiBlast/; https://github.com/DheemanthRegati/cascade_PSI_BLAST. Operating system(s): Platform independent. Programming language: Python. Other requirements: Website: None; Downloadable version: python3, cd-hit module, psiblast, GNU parallel (optional). License: Unlicense. Any restrictions to use by non-academics: No license needed.

Declarations

Ethics approval and consent to participate
The authors declare no competing interests.

Consent for publication
Not applicable.

Competing interests
The authors declare no competing interests.

Received: 27 April 2025 / Accepted: 2 January 2026
Published online: 18 February 2026

References

1. Altschul SF, Madden TL, Schäffer AA, Zhang J, Zhang Z, Miller W, et al. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic Acids Res. 1997;25(17):3389–402. https://doi.org/10.1093/nar/25.17.3389.
2. Zhang Z, Miller W, Schäffer AA, Madden TL, Lipman DJ, Koonin EV, et al. Protein sequence similarity searches using patterns as seeds. Nucleic Acids Res. 1998;26(17):3986–90. https://doi.org/10.1093/nar/26.17.3986.
3. Eddy SR. Profile hidden Markov models. Bioinformatics. 1998;14(9):755–63. https://doi.org/10.1093/bioinformatics/14.9.755.
4. Altschul SF, Koonin EV. Iterated profile searches with PSI-BLAST—a tool for discovery in protein databases. Trends Biochem Sci. 1998;23(11):444–7. https://doi.org/10.1016/S0968-0004(98)01299-5.
5. Bhadra R, Sandhya S, Abhinandan KR, Chakrabarti S, Sowdhamini R, Srinivasan N. Cascade PSI-BLAST web server: a remote homology search tool for relating protein domains. Nucleic Acids Res. 2006;34(suppl_2):W143–6.
6. Kaushik S, Mutt E, Chellappan A, Sankaran S, Srinivasan N, Sowdhamini R. Improved detection of remote homologues using cascade PSI-BLAST: influence of neighbouring protein families on sequence coverage. PLoS ONE. 2013;8(2):e56449.
7. Schäffer AA, Aravind L, Madden TL, Shavirin S, Spouge JL, Wolf YI, et al. Improving the accuracy of PSI-BLAST protein database searches with composition-based statistics and other refinements. Nucleic Acids Res. 2001;29(14):2994–3005.
8. Mott R. Local sequence alignments with stochastic match scores. Bioinformatics. 1999;15(6):455–62. https://doi.org/10.1093/bioinformatics/15.6.455.
9. Pruitt KD, Tatusova T, Maglott DR. NCBI reference sequence (RefSeq): a curated non-redundant sequence database of genomes, transcripts, and proteins. Nucleic Acids Res. 2005;33:D501–4. https://doi.org/10.1093/nar/gki02.
10. UniProt Consortium. UniProt: a hub for protein information. Nucleic acids research. 2015;43(D1):D204–12.
11. Fu L, Niu B, Zhu Z, Wu S, Li W. CD-HIT: accelerated for clustering the next-generation sequencing data. Bioinformatics. 2012;28(23):3150–2.
12. Boeckmann B, Bairoch A, Apweiler R, Blatter MC, Estreicher A, Gasteiger E, et al. The SWISS-PROT protein knowledgebase and its supplement TrEMBL in 2003. Nucleic Acids Res. 2003;31(1):365–70.
13. Murzin AG, Brenner SE, Hubbard T, Chothia C. SCOP: a structural classification of proteins database for the investigation of sequences and structures. J Mol Biol. 1995;247(4):536–40.
14. Burley SK, Berman HM, Kleywegt GJ, Markley JL, Nakamura H, Velankar S. Protein Data Bank (PDB): the single global macromolecular structure archive. Protein crystallography: methods and protocols. 2017:627–41.
15. Tange O. GNU parallel 2018. Lulu. com; 2018.
16. Suzek BE, Huang H, McGarvey P, Mazumder R, Wu CH. UniRef: comprehensive and non-redundant UniProt reference clusters. Bioinformatics. 2007;23(10):1282–8.
17. Rawlings ND, Barrett AJ, Bateman A. MEROPS: the peptidase database. Nucleic acids research. 2010;38(suppl_1):D227–33.

18. Iyer MS, Bhargava K, Pavalam M, Sowdhamini R. GenDis database update with improved approach and features to recognize homologous sequences of protein domain superfamilies. Database. 2019;2019:baz042.

Publisher's note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.