MEST (Multiple EST Screening Tool):  
A web application for automation of BLAST queries of EST libraries

Technical Report: Kevin English\textsuperscript{1}, Xavier Pochon\textsuperscript{2}, Ruth D. Gates\textsuperscript{2} and Guylaine Poisson\textsuperscript{1}

\textsuperscript{1}Department of Information and Computer Science, University of Hawai'i  
\textsuperscript{2}Hawai'i Institute of Marine Biology, University of Hawai'i
ABSTRACT

EST (Expressed Sequence Tag) libraries are an important tool in Gene discovery and phylogenetic analysis. As the number of EST libraries grow, new tools are being made to manage and screen this data. In 1999, the d2_cluster method was developed for clustering ESTs. D2_cluster was used successfully to create the STACKdb human gene index, a database of virtual human transcripts. ReNaBi publishes a set of tools for EST clustering and analysis including EST express, an end user tool that allows users to highlight useful genes and manage cDNA library projects. Other end user tools include ESTin, a tool for building a dbEST submission file from massive EST sequences. EST have been shown to be useful in performing phylogenetic analyses. MEST (Multiple EST Screening Tool), is an EST screening tool that provides a web-based interface for the use of BLAST and ClustalW with the specific goal of finding the closest alignment to perform phylogenetic analysis and design primers. The method of using BLAST to determine similarity as a method of performing phylogenetic analyses has been well established by Roeding. MEST uses new Web 2.0 technologies to achieve the look and feel of a desktop application. The tool was built with inputs from end users (biologists) over the course of one year. While the final tools does not contain all of the features requested by the end users, it contains sufficient functionality to make it a useful contribution to the Bioinformatics community. The framework upon which the tool is built is extensible and can be used for future projects dealing with FASTA Files and BLAST.
Table of Contents

ABSTRACT ........................................................................................................................................2
INTRODUCTION .............................................................................................................................5
WEB APPLICATION DESCRIPTION .................................................................................................8
  User Accounts & Projects ...........................................................................................................8
  Basic Data Elements of MEST ....................................................................................................8
  FASTA Files & Databases ........................................................................................................10
  BLAST ..........................................................................................................................................10
DESIGN & IMPLEMENTATION .........................................................................................................14
  Application Workflow Design ......................................................................................................14
  Database Design ........................................................................................................................15
  Web Application ........................................................................................................................16
  Job Queue ..................................................................................................................................17
  Job Execution Approximation .....................................................................................................18
Testing ............................................................................................................................................18
  Unit & Functional Testing ..........................................................................................................18
Future Work ...................................................................................................................................19
CONCLUSION .................................................................................................................................19

Illustration Index

Illustration 1: The database browsers allows users to organize their Databases in a hierarchical manner........................................................................................................................................9
Illustration 2: To remove duplicate or similar sequences from a database, users can clean the file using output generated from BLAST. A BLAST query is run with the databases as both the source and the target. The BLAST output is parsed, highly similar sequences will be removed and a new 'cleaned' database will be generated........................................................................................................11
Illustration 3: Source FASTA files contain names which are long or confusing. Users may sequentially rename the sequences. Providing a prefix, (e.g. 'XX_'), the sequences will be renamed according to the insertion order (e.g. XX_00001, XX_00002). ..................................................................................................................11
Illustration 4: BLASTing databases against one can be done through a simple dialog box. Basic BLAST parameters are included......................................................................................................12
Illustration 5: BLASTing against the NR/NT database can be accomplished using a dialog box.....13
Illustration 6: Preview ClustalW output in MEST ........................................................................14
Illustration 7: The MEST Workflow involves i) uploading files, ii) extracting sequences into databases, iii) cleaning the database, iv) BLASTing databases against one another, v) BLASTing databases against NR-NT, vi) running ClustalW alignments on generated databases.........................15
Illustration 8: The MEST Entity-Relationship Diagram ...............................................................16
Index of Tables

Table 1: List of Job Classes: These are objects available in the system to perform a background execution function.
INTRODUCTION

A well-resolved phylogenetic tree demonstrating the relationships between species is an important goal in evolutionary biology, and fundamental for comparative studies in biology. Multiple sources of gene sequence data including ESTs are increasingly being used to resolve phylogenetic relationships, and frequently more than 50 genes are being inferred to address key questions about the early evolution of eukaryotes (Burki and Pawlowski 2006; Gadagkar et al. 2005; Phillipe et al. 2005; Rodriguez-Ezpeleta et al. 2007; Ruiz-Trillo et al. 2008; Shalchian-Tabrizi et al. 2008a, 2008b). Recent studies have shown support for the grouping of known eukaryotes into a handful of super groups (Burki et al. 2007; Rodriguez-Ezpeleta et al. 2007; Keeling 2004). The use of multi-gene data in phylogenetic reconstruction (Delsuc et al. 2005; Dutilh et al. 2004) can vastly improve the phylogenetic signal because as the number of genes increases, phylogenetic relationships are better resolved and receive higher statistical support (Burki et al. 2007; Rodriguez-Ezpeleta et al. 2007; Minge et al. 2009).

Phylogenomic work by researchers from the Hawaii Institute of Marine Biology (University of Hawaii) on the symbiotic dinoflagellates (genus Symbiodinium) occurring in corals is in agreement with these observations and clearly demonstrated that the concatenation of nuclear, chloroplastic, and mitochondrial genes resulted in a dramatic increase in resolution and nodes support (Pochon et al. in prep.) It is highly probable that future phylogenetic investigations will increasingly rely on multi-gene datasets rather than single locus reconstructions, and therefore efforts on creating user-friendly bioinformatics tools for thorough, automated, and rapid assessments of genomic databases should be prioritized.

MEST contains a work-flow that has been designed specifically for this phylogenetic research by supporting the discovery of genes and the design of primers. The goals of the MEST project is to provide a small application that can be extended or used as a starting point for other
bioinformatics software developers. MEST is not built from the ground up. It incorporates existing technology including Ruby on Rails\textsuperscript{21}, BioRuby\textsuperscript{22} and a Ruby on Rails plugin for background job execution (delayed\_job\textsuperscript{23}).

With the introduction of web-based e-mail applications, photo and file sharing sites and wiki-style collaboration tools, users have become more comfortable working and collaborating through on-line medium. Wolff, Fitzpatrick & Youssef describe this: “In Web 2.0, the shift from passive to active participation has radically transformed the way we interact and understand how the Web works.”\textsuperscript{24} M.E.S.T (Multiple EST Screening Tool) incorporates this model of Software Development and distribution in its implementation. It exposes the basic options for BLAST and ClustalW and manages the input data (FASTA files) and output data (ClustalW alignment files and FASTA Files) through a hierarchical based work-flow in a web application. The application gives researchers a better collaborative environment by providing a central repository for relevant files for a research project.

In the BioInformatics, BLAST\textsuperscript{25} and ClustalW\textsuperscript{26} are tools that allow for local alignment and multiple sequences alignment respectively. NCBI BLAST\textsuperscript{27} and numerous other bioinformatics programs are hosted on the NCBI and freely available to all researchers. A web-based version of ClustalW\textsuperscript{28} is hosted on EBI. However, while these tools clearly benefit the entire research community, the on-line versions of these tools have some disadvantages. Of particular concern is the fact that users are responsible for managing the storage of their input and output data on their local computers or on network file systems.

A standalone version of BLAST is available for download from NCBI. The basic version contains a command line interface. The BLAST tool contains over 30 command line options. These command line options are complex and require some knowledge of BLAST and its algorithm. To further complicate matters, managing and parsing the output of BLAST commands must be accomplished independently of running a BLAST query. For a project that contains multiple data files, the files can quickly become disorganized and difficult to manage. As there is no central
repository for their FASTA files and the outputs of BLAST queries, end users must rely on local hard drives or other network resources to collaborate. These challenges may inhibit a productive work-flow in a collaborative environment and distract users from their research. For example, a project involving new Symbiodinium gene development required more than a year of work for Xavier Pochon (Hawaii Institute of Marine Biology, University of Hawaii) to successfully investigate ten potential markers. This time could be significantly reduced by distributing work in a collaborative environment, automating BLAST queries and ClustalW alignments and logically organizing their output artifacts.

A standalone version of ClustalW is also freely available for download. The command line version of ClustalW offers similar challenges. For a project that requires the generation of thousands of ClustalW alignment files, the burden of managing these files on a normal file system can quickly overwhelm a research project.

Other tools exist for aligning sequences. Some of these tools are desktop based applications like BioEdit. BioEdit is a sequence alignment editor that works on Windows 95/98/XP. Unfortunately, there is no Macintosh version and there is no web based version. This limits the users' ability to share data and collaborate. Each user must be running Windows and they must install the software.

A popular commercial DNA sequence assembly software is Sequencher. Sequencher is capable of executing DNA assembly algorithms, performing sequence editing, restriction mapping, data import/export features and many other tasks. According to the companies product sheet, “Sequencher is currently used for sequence analysis tasks in every major genomic and pharmaceutical company as well as numerous academic and government labs in over 40 countries around the world.” However, Sequencher is not free software and requires a hardware key license to operate the software.

Another freely available software that allows sequence alignments, genetic distances calculations, and basic phylogenetic analyses, is Mega v4.0. MEGA is an integrated tool for
conducting automatic and manual sequence alignment, inferring phylogenetic trees, mining web-based databases, estimating rates of molecular evolution, and testing evolutionary hypotheses\textsuperscript{32}. It is also only available on Windows.

**WEB APPLICATION DESCRIPTION**

The M. E. S. T. web application provides a simple, Web 2.0 interface for researchers who wish to screen EST files. The interface includes features for cleaning EST files using BLAST, BLASTing EST libraries against one another and performing Multiple sequence alignments.

**User Accounts & Projects**

Each user is given an account in the system. The account consists of a login and a password. Each user is given their own workspace defined as a project. Projects in the system are independent of one another. This gives users the flexibility to upload and process sequence data without overwriting or creating a conflict with data uploaded by other user. Users have access to the projects of other users. This allows users to collaborate on projects, share resources and track the progress the other users are making on their projects.

**Basic Data Elements of MEST**

M.E.S.T contains a fundamental data elements that are exposed to the end user. These data elements are **Database Groups, Databases** and **Sequences**.

A sequence is simply a string of characters. The string will contain characters from one of the biological alphabets: protein or cDNA.

A database has been defined as a collection of related sequences. Each database has a parent database and therefore databases are related to one another though a hierarchical structure. Each
database has a database type. There are different database types in the system and these are mostly transparent to the user. Some of the database types are:

- Raw Databases
- Cleaned Database
- Match Databases

For example, FASTA Files are uploaded and extracted into “Raw Databases.”

Database groups are groups of databases. Typically, the raw databases reside in the root database group and databases generated from BLAST operations against the raw databases are assigned to children database group. Illustration 1 shows a display of the database browser feature.

Illustration 1: The database browsers allows users to organize their Databases in a hierarchical manner.
**FASTA Files & Databases**

Users enter data into the system by uploading FASTA files containing EST Sequence. There is not a limit on the size of the files or the number of sequences allowed in each file. The system has been tested with files of up to 10,000 nucleotide sequences. After FASTA files are uploaded, the user extracts the sequences into databases.

**BLAST**

After sequences are extracted from the FASTA files into a raw databases, BLAST is used to clean a raw database resulting in a new database that will have the database type CLEAN. Cleaning a database is to be thought of as removing redundancies from that database. If the new database contains sequence A, B and C and sequence A is extremely similar to B and C, then we assume that B and C are redundancies and therefore only sequence A will be preserved in our new cleaned database. The cleaning is done by running BLAST and setting the query file (option -i) and the database file (-d) to the FASTA database that has been generated using the formatdb \(^{33}\) command. The user may specify the expected value parameter and other pertinent parameters to the BLAST program to adjust the sensitivity of the BLAST execution. The resulting cleaned database will contain relatively unique sequences.
Often EST source FASTA files contain names which are long or confusing. Thus in addition to having the ability to clean the raw FASTA file, the user may sequentially rename the sequences. Providing a prefix, (e.g. 'XX', 'AA'), the sequences will be renamed according to the insertion order (e.g. XX00001, XX00002, etc.). The original sequence name is stored in the database for historical purposes but any further reference to this sequence will use the new name. Illustration 3 displays the window used by users to rename sequences within a database.

Illustration 2: To remove duplicate or similar sequences from a database, users can clean the file using output generated from BLAST. A BLAST query is run with the databases as both the source and the target. The BLAST output is parsed, highly similar sequences will be removed and a new ‘cleaned’ database will be generated.

Illustration 3: Source FASTA files contain names which are long or confusing. Users may sequentially rename the sequences. Providing a prefix, (e.g. 'XX_'), the sequences will be renamed according to the insertion order (e.g. XX_00001, XX_00002).
Cleaned databases may be used as a "test database". Test databases may be BLASTed against other test databases or a merged group of test databases, referred to in the application as the "target database". The BLAST results will be parsed and a child database group will be generated. The contents of the child database group will be a set of new databases, one for each sequence that contains more than one hit in the target database.

Illustration 4: BLASTing databases against one can be done through a simple dialog box. Basic BLAST parameters are included.

The generated databases in the resulting database group could contain sequences which are potentially identifiable genes. Matched sequences are considered to be sequences in the EST library that often expressed and therefore could represent phylogenetically useful genes. For example, if sequence1 is highly similar to sequence2 and sequence3, it will be inferred that sequence1, sequence2 and sequence3 are specific genes that have been expressed in the library or that they represent only a portion of a gene. The new database will each contain one gene or only a portion of a gene. In the example, sequence1, sequence2 and sequence3 will be in one database. This database
will belong to a database group and will be represented as a subset of the target database in the user interface tree component. To determine if the genes are known, the sequences in the generated databases may be BLASTed against the NR-NT databases. If there are hits in the NR-NT BLAST results, a set number of the NR-NT sequences are appended to the existing database. The user is given the option to limit the number of sequences appended.

To perform further analyses, users can use an alignment tool, such as ClustalW, to generate an alignment file. The alignment file will help biologists identify specific genetic markers (a genetic marker is a gene or a DNA sequence with a known location,) and design primers (a primer is a strand of nucleic acid that serves as a starting point for synthetic DNA replication). The output of the ClustalW job may be previewed in MEST or downloaded and analyzed in a separate program.
In addition to facilitating BLAST, the MEST workbench contains tabs to preview sequences in the database, view the raw text output of a BLAST command, and view the jobs that are in the job queue. Typically, users will submit their job and preview the job queue. The job queue provides time approximation of the job execution length.

**DESIGN & IMPLEMENTATION**

The MEST tools consists of a back-end SQL-database, a front-end Ruby on Rails application and separate job processing queue.

Ruby on Rails has become a popular platform in recent years primarily because of its use of the flexible object-oriented programming language. Like Perl and Python, Ruby is a dynamically typed language. It has the flexibility of a scripting language and at the same time is a top-to-bottom object oriented solution. Like BioPerl and BioPython, a project Bioinformatics library for Ruby has been create called BioRuby. At the time of this paper, BioRuby was on version 1.3.1.

*Application Workflow Design*

The workflow of the application represents the workflow of researchers at Gates laboratory at the Hawai’i Institute of Marine Biology, UH. It was designed with the assistance of Biologists and
a Bioinformatics professor (Dr. Guylaine Poisson.) The workflow involves multiple steps. These steps are highlighted graphically in Illustration 6.

Illustration 7: The MEST Workflow involves i) uploading files, ii) extracting sequences into databases, iii) cleaning the database, iv) BLASTing databases against one another, v) BLASTing databases against NR-NT, vi) running ClustalW alignments on generated databases.

**Database Design**

The design of the database is very simple and is similar to the BioSQL database. The schema for the open source BioSQL database was not used because it does not contain the desired relationships between sequences and databases. The database design for MEST incorporates a Many-to-Many relationship between databases and sequences. This allows sequences information to be reused. It also gives the user the ability to track each sequence back to it's source EST file. Illustration 6 shows a simplified version of the entity-relationship diagram.
The web application was implemented in Ruby on Rails. Ruby on Rails was chosen because of its emphasis on convention over configuration and its use of a model-view-controller design pattern. The MVC pattern is very popular in web application frameworks because it isolates database manipulation objects from presentation logic. In implementing the MEST application, we chose to use the "skinny controller" approach based on a RESTful (Representational State Transfer\(^5\)) web services architecture. The "skinny controller" approach suggest that we put the majority of our back-end logic and database manipulation code in our model objects. Our biggest controller has 54 lines of code whereas our largest model contains 197 lines of code. By using the RESTful architecture, we define all services in our application as resources and have a set vocabulary of how we will interact with them. This brings consistency to our application design.

The web application back-end database is currently MySQL. However, the Ruby on Rails database adapter is database independent and therefore we could easily use another relational database management system (RDBMS) system. In addition to database manipulation, the models are responsible for storing the uploaded FASTA files on server. This is accomplished by using a

Illustration 8: The MEST Entity-Relationship Diagram
Ruby on Rails plugin called Paperclip\(^36\). Our FastaFile model is responsible for saving uploaded files, extracting sequences into the database and generating new FASTA files from raw or generated databases. One initial challenge is to keep the data in the FASTA file in sync with the data in the databases because only FASTA files may be used in BLAST queries.

The web front-end consists of HTML forms and tables that display databases, FASTA files and our job queue. The web applications workbench was implemented using a Javascript toolkit called ExtJS\(^37\). ExtJS is also used by NCBI. It contains many rich front-end components that we utilize like grids, trees and windows. The ExtJS front-end components easily interact with our restful controllers pulling data for the different resources in our application. Some of the challenges of implementing the front-end included developing a consistent, user-friendly interface, debugging Javascript, displaying the large amount of data in the system in a limited screen real estate and notifying the user of status of running jobs.

**Job Queue**

It was determined in the early stages of implementation that execution of BLAST commands and other work intensive functions should not be performed over an HTTP connection, because it would result in unpredictable outcomes. For example, if a user refreshes the page or closes their browser while a large BLAST query is running, the processing will be killed and left in an unfinished state. Therefore, a back-end job queue system was implemented using a Ruby on Rails plugin called Delayed Job. Each time the user requests a BLAST execution, a job is created and placed in the job queued. A separate process monitors the queue and executes jobs in order of job submission. The Delayed Job program can be scaled to have multiple workers, although this features was not used because it could be complicated by execution dependencies. Table 1 shows the current job objects that available to the user.
<table>
<thead>
<tr>
<th><strong>Job Class</strong></th>
<th><strong>Functional Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Jobs::ExtractSequences</td>
<td>Extract Sequences from a file and create a new database</td>
</tr>
<tr>
<td></td>
<td>containing those sequences.</td>
</tr>
<tr>
<td>Jobs::CleanDatabaseWithBlast</td>
<td>Clean a database by BLASTing it against itself and removing</td>
</tr>
<tr>
<td></td>
<td>highly similar sequences.</td>
</tr>
<tr>
<td>Jobs::RenameSequencesInDb</td>
<td>Rename sequences in a database sequentially.</td>
</tr>
<tr>
<td>Jobs::GenerateFasta</td>
<td>Generate a FASTA file from a database.</td>
</tr>
<tr>
<td>Jobs::BlastAndCreateDbs</td>
<td>BLAST 2 databases against each other and create multiple</td>
</tr>
<tr>
<td></td>
<td>databases containing sequences and their matches.</td>
</tr>
<tr>
<td>Jobs::BlastNrNt</td>
<td>BLAST a group of databases against the NR-NT databases</td>
</tr>
<tr>
<td></td>
<td>appending the results to each database.</td>
</tr>
<tr>
<td>Jobs::Clustalw</td>
<td>Perform a ClustalW alignment on database.</td>
</tr>
</tbody>
</table>

Table 1: List of Job Classes: These are objects available in the system to perform a background execution function.

**Job Execution Approximation**

A very simple approximation system was developed to present users with an approximate completion time for their job. This was achieved by looking at previous jobs that were run in the system, averaging the completion times and developing a formula to estimate the future completion time. The current completion time is specific to the machine that MEST is currently running on. A future enhancement to MEST will be to develop a thorough statistical based approximation system. This system would evaluate the execution times for each job in system. From that data it will calculate future execution times.

**TESTING**

Testing was done on MEST in the form of unit testing and functional testing as well as user acceptance testing.

**Unit & Functional Testing**

Ruby on Rails includes a built-in testing framework called Unit:Test. The Unit::Test framework is complimented by the "should" gem, an expressive testing DSL. Unit tests were written for each model and each job library. Functional tests were written for the restful web service interfaces.
Because of the evolving nature of MEST, no integration tests were written. A test coverage tool for Ruby on Rail is available called Rcov\textsuperscript{39}. Rcov executional the project's unit and functional tests and determines which lines of code are tested and which are not. The code coverage of the unit tests and functional tests as determined by Rcov was 75.27%. There is no easy way to test the Javascript front-end.

**FUTURE WORK**

There are many enhancements which can be done to the MEST application including:

- Creating a sign up page for users to request a new account.
- Create more personalized accounts with a social networking aspect. Allows users to share their project with other users in the system and restrict access to read only or read-write.
- Filtering BLAST results retrieved by performing BLAST queries against NCBI by Taxonomy.
- Add auto-refresh to the Workbench panel.
- Enhance the job queue to run more than one job simultaneously. Create a job queue for each user.
- Give the user the option to automatically extract sequences into a database.
- Improve the time approximation system.

**CONCLUSION**

M.E.S.T. demonstrates the power of web-based technologies to be used as a standalone applications. While the application needs improvements, it accomplishes many of the goals that were anticipated including providing an accessible web-based front-end for BLAST and ClustalW. The project establishes BioRuby and Ruby on Rails as a viable platform for BioInformatics.
applications. RESTful web interfaces also have a demonstrated value in this space and will continue to play an important role in evolving technology.


Byungwook Lee. ESTin: A Program for Building a dbEST Submission File from Massive EST Sequences.


Ruby on Rails Home Page: http://rubyonrails.org/


Delayed_job (or DJ) encapsulates the common pattern of asynchronously executing longer tasks in the background. http://github.com/tobi/delayed_job


30 http://www.genecodes.com/sequencher/faq.html#lic1
32 MEGA 4: Molecular Evolutionary Genetics Analysis, http://www.megasoftware.net/
35 Roy Thomas Fielding, Architectural Styles and the Design of Network-based Software Architectures, Chapter 5: Representational State Transfer (REST)
38 Thoughtbot Shoulda: Makes tests easy on the fingers and the eyes, http://github.com/thoughtbot/shoulda
39 Rails rcover: rails_rcover provides easy-to-use Rake tasks to determine the code coverage of your unit, functional, and integration tests, http://agilewebdevelopment.com/plugins/rails_rcover