TESTING THE SPEED OF USEARCH AND BLAT IN COMPARISON TO BLAST AND DETERMINING THEIR SENSITIVITY FOR DETECTING ORTHOLOGS AS RECIPROCAL BEST HITS

A Thesis Submitted to
Wilfrid Laurier University

by

Natalie Ward

In Partial Fulfillment for the degree of
Bachelor of Science (Honours)
in the Department of Biology

Waterloo, ON

© Spring 2012

Abstract

Generating accurate alignments while searching through increasing amounts of genomic sequence data within a reasonable amount of time poses a challenge current database search programs. As a result, two relatively new sequence alignment algorithms USEARCH and the Blast-Like Alignment Tool (BLAT), which have been marketed based on their speed and accuracy, were tested to determine if in fact the increased speed offered negatively affects their sensitivity and accuracy. Default command lines specific to each algorithm were run on a UNIX server, specifying *Escherichia coli* K12 as the query and *Bacillus subtilis* as the target. BLAT was the fastest algorithm, finding homologs 104x faster than it took BLASTP to explore the same database; however, BLAT did not return the greatest number of putative homologs. USEARCH found homologs 30x faster than BLASTP. The BLASTP algorithm located the greatest number of potential homologous sequences, finding 5x and 17x more matches than BLAT and USEARCH, respectively, but took the greatest amount of time to do so. Subsequently, different combinations of USEARCH options were specified to determine if increasing the number of target sequences tested also provides greater sensitivity; which option combinations improved the detection of homologs and orthologs was then identified. A program written in PERL was used to run different combinations of –maxaccepts and –maxrejects, testing the *E. coli* K12 query against nine other microorganism targets. For every genome tested, it was observed that a greater number of putative homologs and ensuing orthologs were detected by USEARCH as the number of target sequences tested was increased. Further, specifying a –maxaccepts value of 100 and a –maxrejects value of 1024 gave the user a balance between speed and sensitivity. BLASTP was still able to detect a larger number of homologs and orthologs, however. The quality of the ortholog data was then estimated using an alternate PERL program. Results indicated that the default –maxaccepts and –maxrejects combination is the least accurate with regards to finding true orthologs as the error rate associated with this option was high. Additionally, the error rate for each combination decreased when more target sequences had been tested. These results demonstrate that speed compromises sensitivity but in the case of USEARCH, sensitivity and accuracy, can be improved by changing the value of –maxaccepts and –maxaccepts options.

Acknowledgements

I would like to thank Dr. Moreno-Hagelsieb for providing me with the opportunity to learn about bioinformatics and to become familiar with the computational side of biology previously unbeknownst to me before completion of this thesis. I appreciated his willingness to explain and then re-explain concepts to me. Thank you also to Marc Del Grande, who helped me learn about commands used in the wonderful world of UNIX.

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Database search algorithms

Many different tools have been introduced over the last 40 years to efficiently analyze and compare genetic sequence information: dynamic programming algorithms such as Needleman-Wunsch (Needleman and Wunsch, 1970) and Smith-Waterman (Smith and Waterman, 1981) avoid exploring non-optimal alignments while guaranteeing the user the mathematically highest scoring alignment (Eddy, 2004); search algorithms such as the Basic Local Alignment Search Tool (BLAST) (Altschul et al., 1990, 1997) and FASTA (Pearson and Lipman, 1988) save time and offer accuracy, employing heuristics to find DNA or protein sequences that are similar in base or amino acid composition. It is the job of these algorithms to sort through copious amounts of sequence data spanning hundreds of species and reliably find those target database sequences that share regions of similarity to a particular query sequence of interest. Specifically, these search strategies identify potential homologous sequences, those that share a relationship of common descent (Fitch, 1970).

BLAST is a widely used heuristic algorithm used to locate prospective homologs due to its sensitivity and speed (Edgar, 2010). Involved in its method are two key steps, which are shared among other database search programs that employ heuristics. Initially, a search stage locates regions of high similarity between the query and target sequences (Kent, 2002). Subsequently, an alignment stage scores each of the predicted alignments and assesses which identified regions show the highest degree of similarity according to defined threshold constraints (such as a maximum E-value), with resultant optimal alignments being produced (Kent, 2002). It is through this location of similar genetic sequences, or putative homologs, that one can determine the hypothetical relatedness of various species, the potential function of an uncharacterized protein, or the identification of a gene in a newly sequenced genome (Zvelebil and Baum, 2008).

However, identifying the optimal alignment for the query sequence of interest amongst a whole database of sequences is becoming more time consuming. Biological sequence data present in these databases has been accumulating since the introduction of sequencing techniques in the 1970's and continues to increase (Edgar, 2010). Dynamic programming algorithms would be an inefficient tool to search the databases as they are

rigorous and time consuming; the amount of time it takes BLAST to find similar sequences is comparative to the length of the query sequence as well as the size of the database being used (Altschul et al., 1997). Consequently, currently employed algorithms are becoming progressively more unproductive and their accuracy may be diminished to decrease the overall time it takes to find homologous sequences (Altschul et al., 1997).

To solve this problem, two recently introduced algorithms, the BLAST-like alignment tool (BLAT) (Kent, 2002) and USEARCH (Edgar, 2010), have been created to decrease the amount of time it takes to find significant alignments between a query sequence and a target database of sequences, while preserving accuracy; each program employs various strategies to achieve this.

1.1.1 USEARCH

USEARCH is a sequence analysis tool used to search sequence databases for homologous targets to a particular query sequence of interest. It was developed by Robert Edgar and was released in 2010 for use. USEARCH executes many different algorithms including UBLAST, a database search algorithm; UCLUST, used for clustering; and UCHIME, used to find chimeras in a sequence (Edgar, 2010). The UBLAST algorithm has been shown to find alignments at a much faster speed than BLASTP, while maintaining comparable sensitivity: using 1000 query sequences from each of the Pfam and Rfam databases with the remainder of the protein and RNA sequences, respectively, used as the target sequences, UBLAST was ~350x faster at locating potentially homologous sequences than BLASTP and had similar sensitivity above 20-35% sequence identity (Edgar, 2010).

There are similarities between the alignment methods used by BLAST and USEARCH: both use k-mers of specific lengths to find high scoring pairs (HSPs) and both use similar heuristic algorithms to increase computational speed (Edgar, 2010). Despite this, there are several methods and parameters incorporated into the USEARCH program which decrease the amount of CPU time it takes to find optimal homologous matches between a query and a target sequence.

When posed with a query sequence, USEARCH ends the search stage when the first hit, or a good match, is found (Edgar, 2010). It does this by testing database sequences in order of decreasing number of short amino acid or nucleotide segments in common with

a query sequence of interest, termed U-sorted order (Edgar, 2010). The rationale here is that sequences having a similar composition are likely to have a greater number of words, or short fragments in common and as a result, a very good hit or even the best hit will be the first found (Edgar, 2010). As an option, the search for highly similar target sequences can be terminated by USEARCH after a threshold number of potentially homologous sequences, or "accepts" have been located; this is set by the –maxaccepts option and the default is one. Further, after a specified number of failed attempts to align a target sequence have been reached, the search stage will end. The number of sequences that can be "rejected" before termination is set by the –maxrejects parameter, where the default value is 32.

Together, these parameters used by the USEARCH algorithm allows for a faster detection of homologous sequences by either reducing the amount of time it takes to search a sequence database for homologs, searching only a fraction of the target sequences in the database, or by effectively eliminating sequences in the search stage to avoid subjecting them to the time consuming alignment stage (Edgar, 2010). A variety of default search strategy settings (aside from the ones listed here), such as the search order and word counting methods, can be manipulated to increase sensitivity or to emulate those used in BLAST but as a consequence, speed may be compromised (Edgar, 2010).

1.1.2 BLAT

Developed by W. James Kent in 2002, BLAT is an mRNA, DNA and translated protein alignment algorithm frequently used to generate human expressed sequence tag (EST) and mRNA alignments (Kent, 2002). BLAT is similar to BLAST in that it searches for short matching segments and then extends these hits in both directions to form high-scoring pairs (HSPs) (Kent, 2002), but the search method BLAT employs is the opposite of BLAST. To decrease the time it takes to search for homologous sequences, BLAT generates an index of non-overlapping k-mers from the database instead of generating an index of the query, while at the same time eliminating words from the index that occur frequently (Kent, 2002). In so doing, the program only has the shorter query sequence to analyze for matches instead of sorting through all sequences in the database (Kent, 2002). Indexing the database is the main method that contributes to BLAT's speed.

Additionally in the search stage, the requirements for generating an alignment have been manipulated: BLAT does not require perfect matching k-mers for an alignment to be detected but instead, a mismatch of one letter can produce a hit between a query and target as well as several exact matches within a certain distance from each other in the sequence are considered to indicate homology as they can be joined together to generate a longer homologous sequence alignment (Kent, 2002). It is thought that by allowing a mismatch in a longer word length or by permitting multiple nearby matches, the reliability of the alignment results will be greater than if only perfect matches consisting of small word sizes were tolerated. Both of these strategies decrease the number of potential homologous sequences that are passed on to the alignment stage where it is determined if their similarity exceeds a given threshold (Kent, 2002). As a result, an optimal alignment can be generated with greater speed. The similarities and differences in approach to finding optimal alignments between BLAST, USEARCH, and BLAT are listed in Table 1.

Table 1- Comparison of the default methods used by the alignment programs BLAST, USEARCH, and BLAT.

	Database Search Tool		
Search Parameters	**BLAST**	**UBLAST**	**BLAT**
Database search method	Builds an index of query and uses scoring matrix to find high-scoring matching words for each query word -scans through sequence database for exact matches to word list -all database sequences tested and all potentially homologous sequences reported	U-sort method -search terminated after searching a threshold number of accepts or rejects or when first hit found -rejects a target if it has too few words in common with query	Builds an index of sequence database and scans through query -searches for almost perfect matches (one letter mismatches) and multiple perfect matches
Alignment method	Triggers an extension for each word match -extends these in HSPs -extracts maximal scoring pairs (MSPs); these are used to seed a dynamic programming calculation -determines their statistical significance -each region of homology is reported as separate alignments	Begins by finding HSPs -exact word matches are extended using a BLAST-like algorithm -similarity of HSPs is determined; discarded if don't meet threshold -regions are aligned using banded dynamic programming	Triggers extensions on an unspecified number of perfect matches or hits where one letter mismatches -for protein alignments, extends matches into HSPs -dynamic program finds MSPs - regions multiple perfect matches joined together into longer alignment
Alignment type	Local	Local	Local
Search order	Finds all possible hits in database	Finds one strong hit	Finds all hits in query
Word counting	k-mers	k-mers	k-mers
Word size	Amino acids = 3 Nucleotides = 11	Amino acids = 5 Nucleotides = 8	Amino acids = 5 Nucleotides = 11

1.2 Orthology and It's Working Definition

Database search algorithms locate potentially homologous sequences in a database based on the degree of similarity between a query sequence and several target sequences. However, there are different subtypes of homology based on the evolutionary history of the homologous sequences; identifying the subcategory to which the matched query-target sequences belong is important for establishing, more specifically, the extent of their relatedness.

Orthology is a specific type of relationship between two homologous genes that was established through the separation of a single ancestral gene in the most recent common ancestor after a speciation event (Fitch, 1970). Genes that fit this definition are presumed to have similar or identical molecular biochemical functions in their respective organisms (Kristensen et al., 2011), although this is not always the case (Bork et al., 1998). Extra-paralogs, on the other hand, are genes that diverged after a duplication event and are further separated by a speciation event (Janga and Moreno-Hagelsieb, 2004) and do not retain their original function in different organisms (Kristensen et al., 2011).

Ortholog detection is important for understanding evolutionary conservation or variation of molecular sequences or even how genes are obtained or lost by an organism (Kristensen et al., 2011). In this way, ascertaining which homologous sequences demonstrate orthology allows one to gain insight into evolutionary histories and functional relationships of genes and apply this knowledge to homologous genes whose background is unknown (Kristensen et al., 2011). Therefore, detecting orthology is important if one wishes to determine phylogenetic relationships between sequences, beyond the less informative definition of homology, and reveal which ones share a high degree of relatedness. Figure 1 shows homologous relationships that arise during the course of evolution.

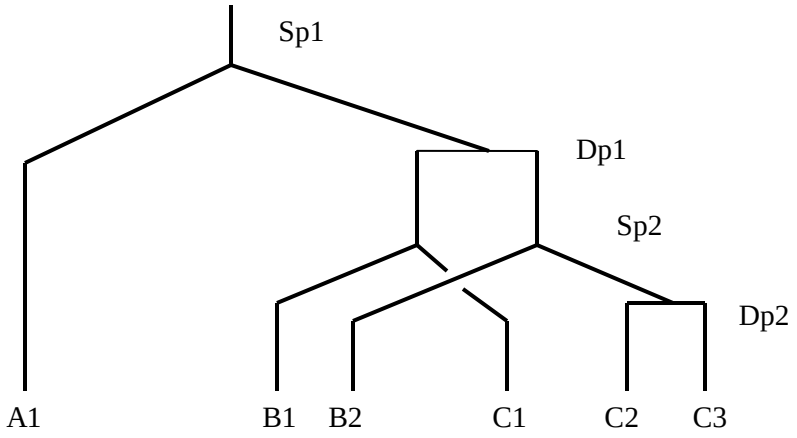


Figure 1- An evolutionary tree depicting the path of events that give rise to orthologous and paralogous genes. Two speciation (Sp1 and Sp2) and two gene-duplication (Dp1 and Dp2) events are depicted. Orthologous genes have a common ancestor at Sp1 or Sp2; paralogous genes have a common ancestor at Dp1 or Dp2. In the case of orthology, C2 and C3 are orthologous to B2 as well as A1 because their common ancestor is located at a speciation event. Genes B1 and C2 are extra-paralogs as their common ancestor resides at duplication event 1 and they are each found in different genomes. Adapted from Fitch (2002).

Ortholog detection can occur once homologous sequences have been established. Phylogenetic tree construction using putative homologs is a commonly employed method to determine orthology (Kristensen et al., 2011). Analysis of the tree would reveal paralogous genes, those having diverged after a duplication event, which cluster more closely together with organisms of the same species as well as ortholgous genes, those that group with organisms of different species (Kristensen et al., 2011). However, despite the appropriateness of the phylogenetic tree procedure to detect orthologs, they require much computational power to generate when considering hundreds of genes in a variety of different organisms. As a result, other methods for orthology detection have been devised. One of these methods uses a short-cut or working definition of orthology termed reciprocal best hits (RBHs), where two genes in two different genomes are deemed to be orthologous if their respective protein products are each other's best hit (Moreno-Hagelsieb and Latimer, 2008).

More specifically, sequences s_1 and s_2 in genomes G_1 and G_2, respectively, are thought to be orthologous if:

(a) s_2 is the best hit, or has the greatest degree of similarity, to s_1 when s_1 is queried against G_2, and

(b) s_1 is the best hit to s_2 when s_2 is queried against G_1 (Wall et al., 2003).

However, if when s_2 is run against G_1, a third sequence is reported as being the best possible RBH, it can be concluded that s_1 and s_2 are not orthologous; both sequences need to be each other's best hit. Figure 2 demonstrates these relationships.

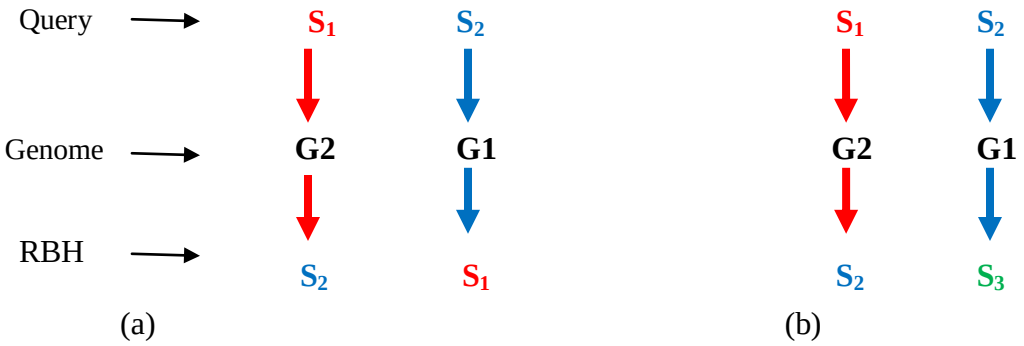


Figure 2- The commonly used working definition of orthology using reciprocal best hits. (a) Two sequences are said to be orthologous if they are each other's best hit when a search is conducted in both directions. (b) If a third sequence (s_3) is found to be the most similar to s_2, it can be concluded that s_1 and s_2 are not orthologs.

1.3 Objectives

During the course of this work, several objectives are to be investigated. Initially, the speed of USEARCH and BLAT, in relation to BLAST, will be determined using the genome of *Escherichia coli* K12 MG1655 as the query and *Bacillus subtilis* as the target. Additionally, their sensitivity, that is, the number of results the algorithm generates, will be analyzed based on the number of homologous sequences they detect. Because USEARCH is user-friendly, as various parameters can be manipulated, the effect of modifying command lines to specify increasing combinations of –maxaccepts and –maxrejects on the speed and sensitivity of the algorithm will be explored using *E. coli* as the query and 10 different microorganism genomes as the targets. From this, optimal parameters (that is, those that strike a balance between speed and sensitivity) to indicate when using USEARCH, will be reported. RBHs will be used to detect orthologous sequences found by

USEARCH, BLAT and BLAST following a database search, to further measure the sensitivity of these algorithms through their ability to locate closely related sequences. Finally, the quality of the RBHs will be examined.

1.4 Hypotheses

It is hypothesized that, even though BLAT and USEARCH can search sequence databases with increased speed and method efficiency, their sensitivity will be compromised as a result. A reduced number of matches, both homologous and orthologous in nature will be reported, when compared to the number extracted by BLAST. Moreover, it is thought that by increasing the number of target sequences USEARCH tests before search termination, sensitivity, as well as the time it takes to locate an alignment, will also increase. However, the speed of USEARCH will still be greater than that of BLAST. Due to BLAST's known sensitivity, it is believed that BLAST will detect a greater number of orthologs than either USEARCH or BLAT, even after quality has been accounted for.

2 Materials and Methods

2.1 Genomes

In the initial experiments, the genome of *Escherichia coli* K12 substrain MG1655 (Blattner et al., 1997) was used as the query of interest and was compared to the genome of *Bacilius subtilis* (Kunst et al., 1997). This was followed by querying the *E. coli* genome against the genomes of nine more microorganisms. They were chosen based on their varying evolutionary distances from *E coli*. Table 2 gives a listing of genomes used. The genomes were chosen from a local protein database held by the Laboratory of Computational Genomics at Laurier.

Table 2- A listing of the 10 microorganisms used as targets and their respective phylogenetic classification. Information pertaining to classification was retrieved from the NCBI Taxonomy Database.

Organism	Domain	Phylum
Salmonella enterica serovar *Typhimurium* (McClelland et al., 2001)	Bacteria	Proteobacteria
Bacillus subtilis (Kunst et al., 1997)	Bacteria	Firmicutes
Streptomyces flavogriseus	Bacteria	Actinobacteria
Flavobacterium johnsoniae (McBride et al., 2009).	Bacteria	Bacteroidetes
Fusobacterium nucleatum (Luca et al., 2011)	Bacteria	Fusobacteria
Nitrobacter winogradsky (Starkenburg et al., 2006)	Bacteria	Proteobacteria
Bordetella pertussis serovar *Tohama* (Parkhill et al., 2003)	Bacteria	Proteobacteria
Treponema pallidum serovar *Pallidum* (Fraser et al., 1998)	Bacteria	Spirochaetes
Methanocaldococcus jannaschii (Bult et al., 1996)	Archaea	Euryarchaeota
Saccharomyces cerevisiae (Bussey et al., 1995, 1997; Feldman et al., 1994; Oliver et al., 1992; Jacq et al., 1997; Dietrich et al., 1997; Murakami et al., 1995; Tettelin et al., 1997; Johnston et al., 1994, 1997; Churcher et al., 1997; Calibert et al., 1996; Dujon et al., 1994; Bouman et al., 1997; Philippsen et al., 1997; Dujon et al., 1994)	Eukarya	Ascomycota

2.2 Algorithm Command Lines

The underlying UNIX system of a MacPro Apple server was used as a platform to run the command lines specific for USEARCH, BLAT, and BLAST. (An example of the command line structure for each algorithm is shown in Table A2.1 in the Appendix). For each of the three algorithms, the default parameters can be changed or inactivated so as to make the program more tailored to fit the user's needs. USEARCH is particularly malleable in this respect. For example, the fields present in the output file can be specified by the "-userfields" option in USEARCH. Additionally, the sensitivity of homolog or

ortholog detection can also be increased through manipulation of the USEARCH command lines, such is seen when the –maxaccepts and –maxrejects options are changed so as to test a greater number of target sequences before a search is ended.

Both USEARCH and BLAT have similarities to BLAST in the way a sequence database is searched (refer to Table 1). By altering the options specified in their respective command lines, one is able to make USEARCH and BLAT run in a similar manner as BLAST. Further, the format of the output file can be changed to resemble one given by BLAST. For example, this can be done by indicating the "–out=" option in BLAT, followed by the "blast9" parameter which indicates the output file be in a NCBI BLAST tabular format. USEARCH also allows one to generate a BLAST-like file; however, in this case, a user-defined output was indicated using the "–userfields" option. A particular expectation value, or E-value, can also be denoted. When running USEARCH and BLAST, an E-value of 1×10^{-6} was specified so as to accurately determine which microorganism protein sequences were truly homologous to those in *E. coli*.

2.3 Detection of Homologous Sequences

Compared were the protein sequences encoded in the *E. coli* K12 genome against the proteins encoded in the *B. subtilis* genome when determining the relative speed and sensitivity of both USEARCH, BLAT, in relation to BLAST as well as when testing various USEARCH parameters. Both *E. coli* and *B. subtilis* are well annotated model organisms and were therefore used in this work. *B. subtilis* was the only genome used as the target at first in order to obtain a sense of the speed and number of hits returned by the algorithms. Subsequently, the *E. coli* genome was queried against the genomes of 9 other microorganisms to test combinations of USEARCH –maxaccepts and –maxrejects parameters and to determine the number and quality of orthologs detected by each program. Figure 3 outlines the various steps taken during the course of this work.

At the outset, default command lines were devised and used to compare the speed of the three database search tools. Subsequently, various USEARCH command lines were synthesized by simply altering the number of –maxaccepts and –maxrejects specified. The commands were run five times with their respective algorithm on the UNIX operating system and the number of potential homologous sequences generated by each program was recorded; the amount of time it took for each to generate an output was also measured.

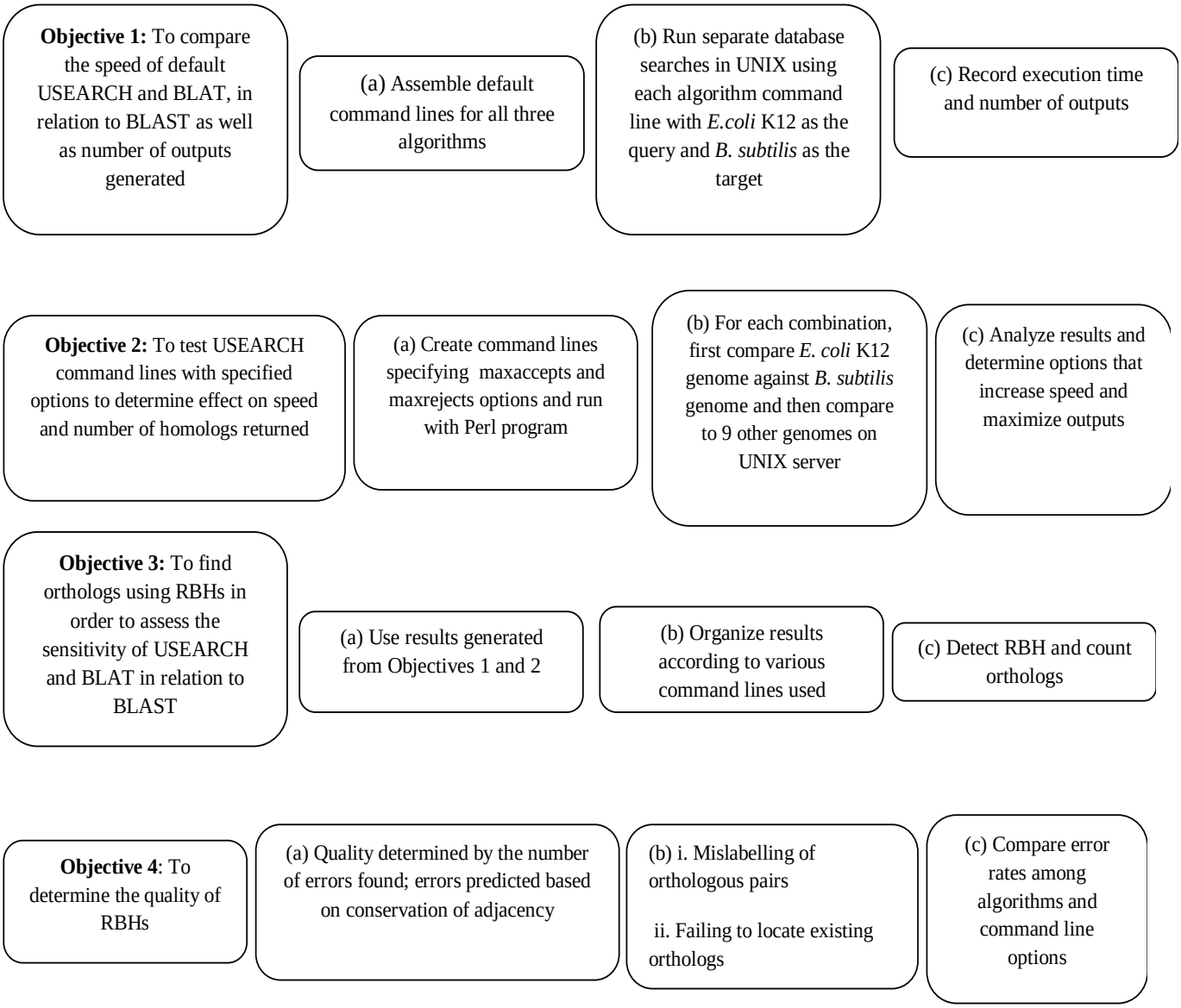


Figure 3- A brief summary of the objectives and the corresponding procedures used to fulfill them are shown. Each diagram is read left to right.

2.4 Testing USEARCH Options

USEARCH was run with various combinations of –maxaccepts and –maxrejects to find out the impact on sensitivity and speed of the algorithm when the number of target sequences tested was increased and how the number of results reported compared to those given by BLASTP. The results were then analyzed and optimal parameters to specify so speed and sensitivity could be maximized, were determined. Full USEARCH command lines were

created and then a PERL program was written by G. Moreno-Hagelsieb, which allowed USEARCH to run different combination sets of –maxaccepts and –maxrejects while monitoring the amount of time and the number of matches (that is, potential homologs) detected. The program was executed on the UNIX server. To begin with, the *E. coli* K12 genome was queried against the genome of *B. subtilis*. Following this, 10 microorganism genomes, including *B. subtilis*, were used as the targets. The same PERL program was used when testing different target genomes; however, the other nine genomes to be tested were specified. An example of a PERL program used is shown in Figure A2, located in the appendix.

2.5 Detection of Orthologs Using RBHs

The data given by USEARCH, BLAT and BLAST is solely a list of homologous sequences found between the query and target genomes. However, homologs can be further subdivided into orthologs, paralogs, and xenologs; of interest here was the detection of orthologs as they provide information as to which sequences are evolutionarily closely related. Therefore, after homologs were detected by the various USEARCH command lines, and by BLAT and BLAST, orthologs were identified. This was achieved by employing two different PERL programs. One of the programs searched USEARCH, BLAT and BLAST homolog results and chose protein matches that showed the greatest similarity based on their bit-score. Orthologs were then detected by determining the number of RBHs. The selected sequences were queried against the genome of their matched target and the number of orthologous sequence pairs found between *E. coli* K12 and each of the 10 microorganisms was recorded by a second PERL program. To be counted as an ortholog, the sequence pairs could not surpass a threshold expectation value (E-value) of 1×10^{-6}, so as to eliminate the possibility of false-positive results. The E-value is a statistical measure used to assess the similarity of two sequences, where the smaller the E-value, the greater the chance two matched sequences share true relatedness (Zvelebil and Baum, 2008). After running each PERL program, the number of orthologous sequences found for each target genome was matched with the corresponding command line that was run to generate those results. The data obtained for each algorithm was then compared based on the number of ortholog outputs given.

2.6 Determining RBH Quality

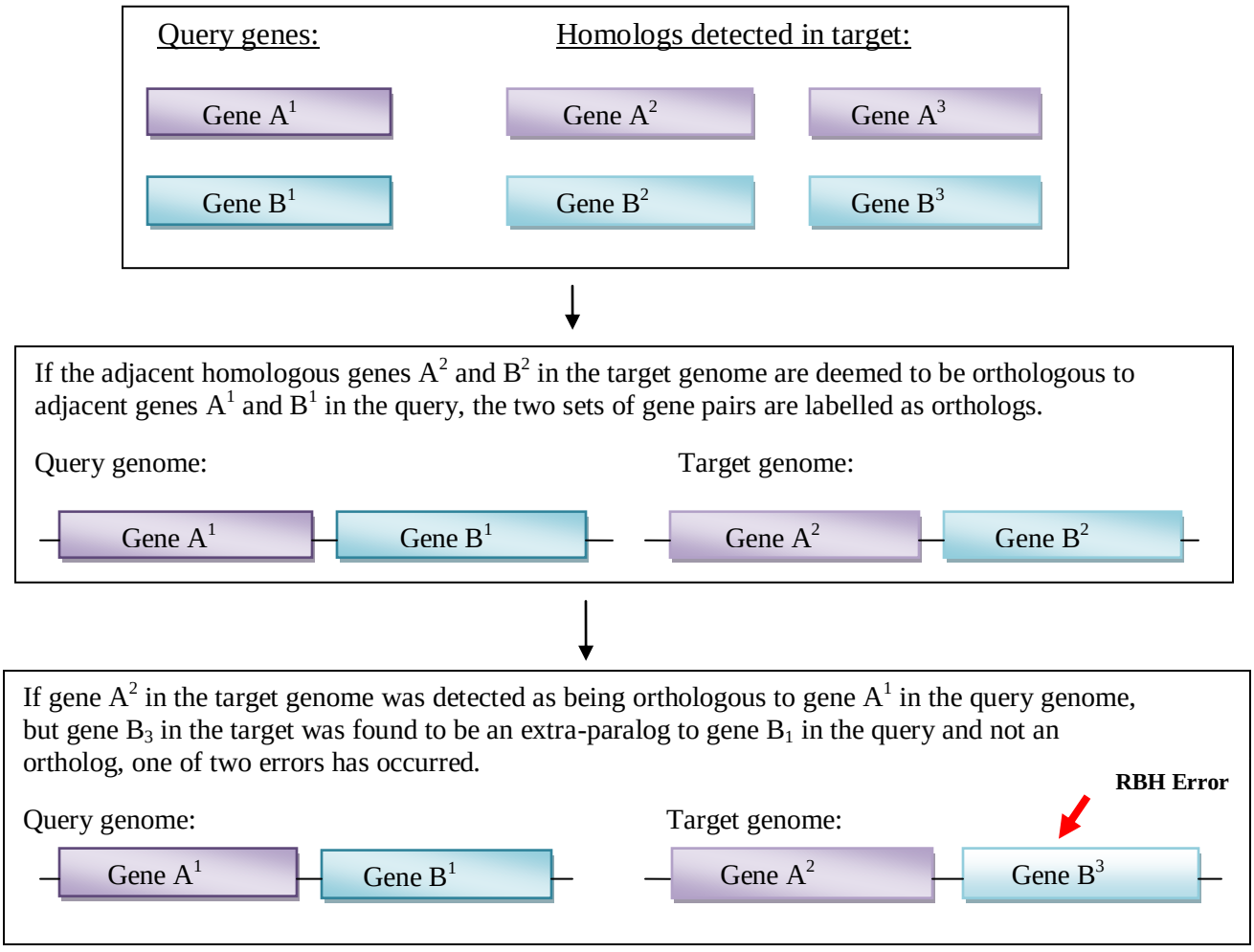
One of the challenges of detecting orthologs is being able to accurately identify and distinguish them from other sequences that share a different type of homologous relationship, such as paralogy (Koonin, 2005). Due to events occurring through evolution, such as horizontal gene transfer, gene loss and gene fusion (to name a few), pinpointing which genes are orthologous can be challenging under the best circumstances (Kristensen et al., 2011). As a result, mistakes can be made when running a program to locate orthologs. To increase the accuracy of the RBH data generated and assess the quality of ortholog detection, the number of errors made while locating orthologous sequences was estimated and those matches that were not deemed true orthologs were eliminated from the total number identified by each algorithm. There are two different situations that were regarded as mistakes: one being the mislabelling of pairs and the second being the failure to locate true orthologs.

Conservation of adjacency was the tool used to assess whether or not two sequences were truly orthologous and is based on comparing the gene order in the target sequence to that in the query. It is supported by the observation that organisms who share a close evolutionary history maintain a similar genetic sequence and therefore, should have genes whose adjacency has been conserved (Moreno-Hagelsieb et al., 2001). However, since gene order between Prokaryotes and Eukaryotes is not usually conserved, *Saccharomyces cerevisiae* was not included in this test.

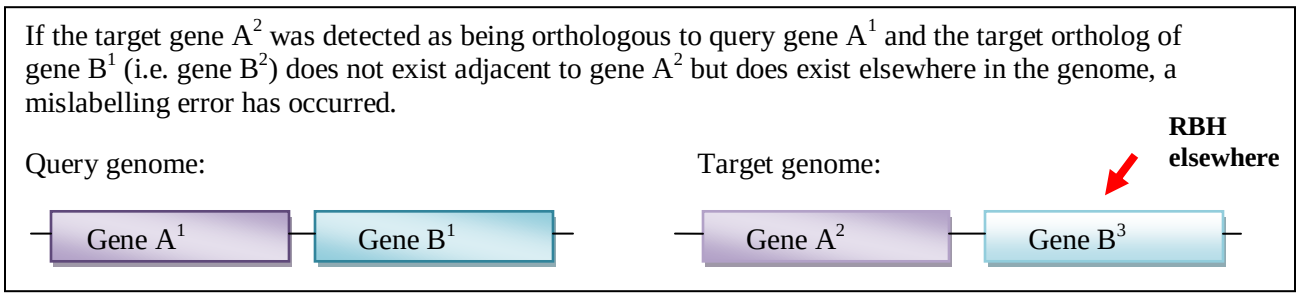
To estimate errors in orthology, G. Moreno-Hagelsieb slightly modified a PERL program to accommodate the data being tested in this work. The program is designed to analyze the gene order in the target and compare this to the adjacent genes in the query genome. Upon detection of contiguous homologs, the program determines if these genes were originally labelled as being orthologous when compared to the query and if this is the case, they are considered to share true orthology. Conversely, if one gene was labelled as an ortholog but the gene next to it was deemed to be an extra-paralog, the program recognizes this as an error. The number of errors is then subtracted from the total number of orthologs originally detected by the particular algorithm.

Additionally, failing to locate existing orthologs can occur when either a paralog in the target genome is misidentified as an ortholog to the query or when an ortholog was

mistakenly deemed a paralog but no matching ortholog could be found in the data returned. In this case, it is concluded that a genuine ortholog was unsuccessfully located during the search stage. The conservation of gene order concept and the ortholog errors that can be detected are shown diagrammatically in Figure 4.



Error 1:



Error 2:

If gene A^2 in the target genome was detected as being orthologous to gene A^1 in the query genome, but no ortholog to gene B^1 in the query was found adjacent to gene A^2 in the target and there does not exist an ortholog to B^1 elsewhere in the genome, a missing ortholog error has occurred.

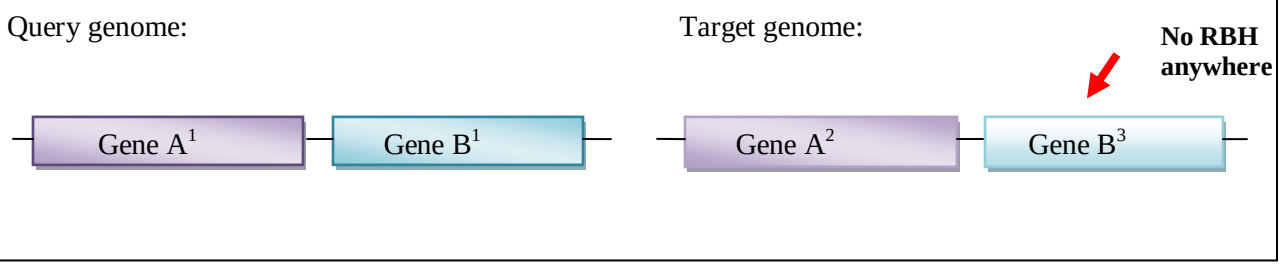


Figure 4- The conservation of adjacency concept used to detect putative orthologs and the mistakes that can occur during the identification of orthologous genes. Initially, homologs to query genes are found by the USEARCH or BLASTP algorithms and then a PERL program checks for the conservation of adjacency of the homologs. If gene adjacency has been conserved, and they were deemed to be query orthologs by the algorithm, the pair of genes is considered to be true orthologs. However, if gene B^2 was deemed to be an extra-paralog of gene B^1, one of two errors has happened: a mislabelled pair or the failure to detect an existing ortholog.

The number errors made by the database search algorithms, when their respective command lines had been run, were recorded by a second PERL program. Error rates were then calculated using the following equation:

$$\text{Error rate} = \frac{N_p}{N_p + N_o}$$

where Np is the total number of errors (mistaken orthologs + undetected orthologs) and No is the correct number of orthologs found (that is, orthologs adjacent to a matching orthologs, or the ortholog adjacent to a paralog).

3 Results and Discussion

3.1 Assessing the Relative Speed and Sensitivity of USEARCH and BLAT

To obtain an understanding of the general speed and sensitivity of USEARCH and BLAT, default command lines for these database search algorithms, as well as BLASTP, were run in UNIX. *E. coli* K12 was the specified query genome (and was the query for the entirety of the project) and was compared to the target genome of *B. subtilis*. The amount of time and the number of matches (i.e. number of potential homologs) generated by USEARCH and BLAT were compared to the data given by BLAST. Figure 5 shows these results. As can be seen, BLAT was the fastest at detecting putative homologs, yielding an output in an average time of 1.5 seconds. USEARCH was also much faster than BLASTP, generating a list of homologous sequences in approximately 5.2 seconds; this increased speed over BLAST was to be expected. However, despite the fact that it took BLASTP an average of 157.8 seconds to locate matches between the target and query, this algorithm was able to provide the user with a considerably higher number of matches than that provided by either USEARCH or BLAT. This initial result indicated that although USEARCH and BLAT required less time to find homologs within a sequence database to a query of interest, BLAST offered superior sensitivity by returning a greater amount of results.

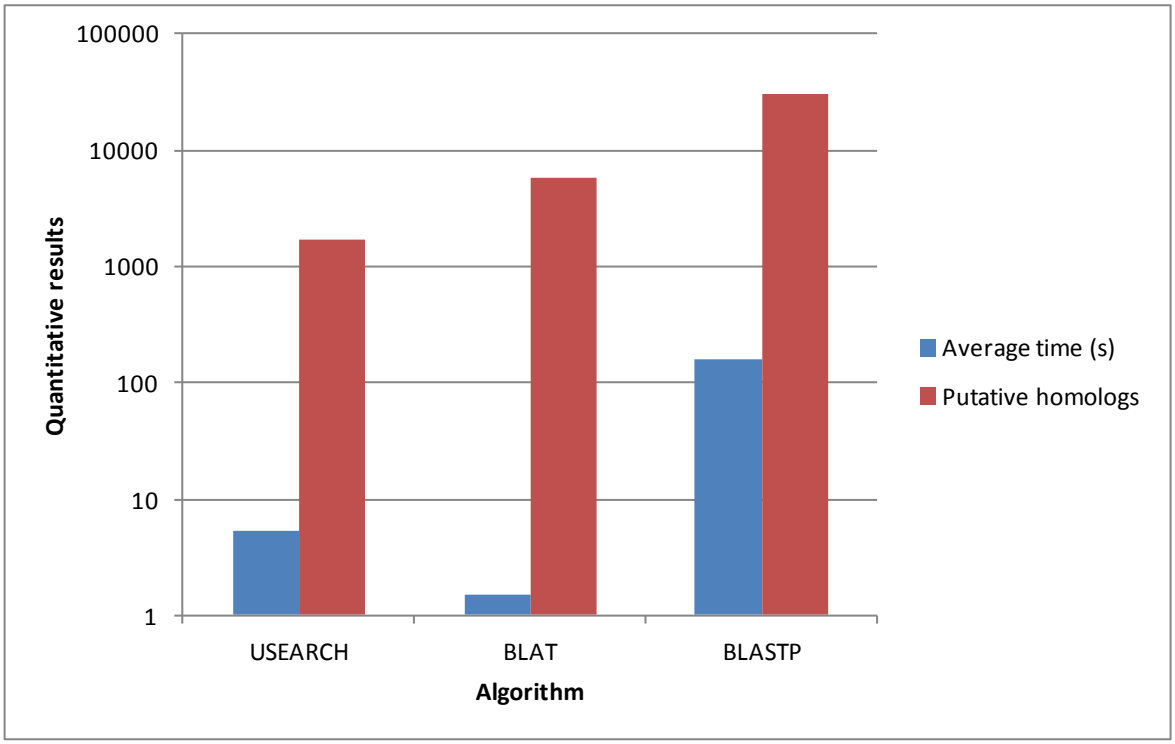


Figure 5- The speed and sensitivity of USEARCH, BLAT and BLASTP. USEARCH and BLAT were run using their default values on a UNIX server and the time it took to generate an output was recorded and compared to the time it took BLAST to do the same. In addition, the number of outputs generated by each algorithm was determined to assess sensitivity. Each command line was run five times and the average time taken was recorded. The number of homologs detected does not vary for each algorithm and the time difference was very small; the standard error was between 0 and 0.001 for time and therefore, was not shown. Results are shown on a log scale for comparison purposes.

3.2 Effect of USEARCH Options on Speed and Sensitivity

3.2.1 *Bacillus subtilis* as the target

3.2.1.1 Speed and Homolog Detection

Since USEARCH allows the user to change and specify many different parameters, further investigation was completed into the effect on the speed and sensitivity of the algorithm by increasing the number of target sequences tested before the search was terminated. This was achieved by testing different combinations of –maxaccepts and –maxrejects, indicating the number of matching and dissimilar targets allowed, respectively, before a search is stopped. Initially when exploring this objective, *B. subtilis* was the only genome used as the target so as to attain a precursory understanding of the consequences associated with altering the default –maxaccepts and –maxrejects parameters. Genomes of nine other

microorganisms were tested subsequently. Command lines specifying the different combinations were run on UNIX using a PERL program; the time and number of probable homologs attained with each command line were recorded. Additionally, these results were compared to the speed and number of matches yielded with the default command lines of both BLAT and scores were normalized against the corresponding data returned by BLASTP. Figure 6 illustrates the effect on time and Figure 7 demonstrates the effect on homolog detection when the number of *B. subtilis* target sequences tested against the query was gradually increased.

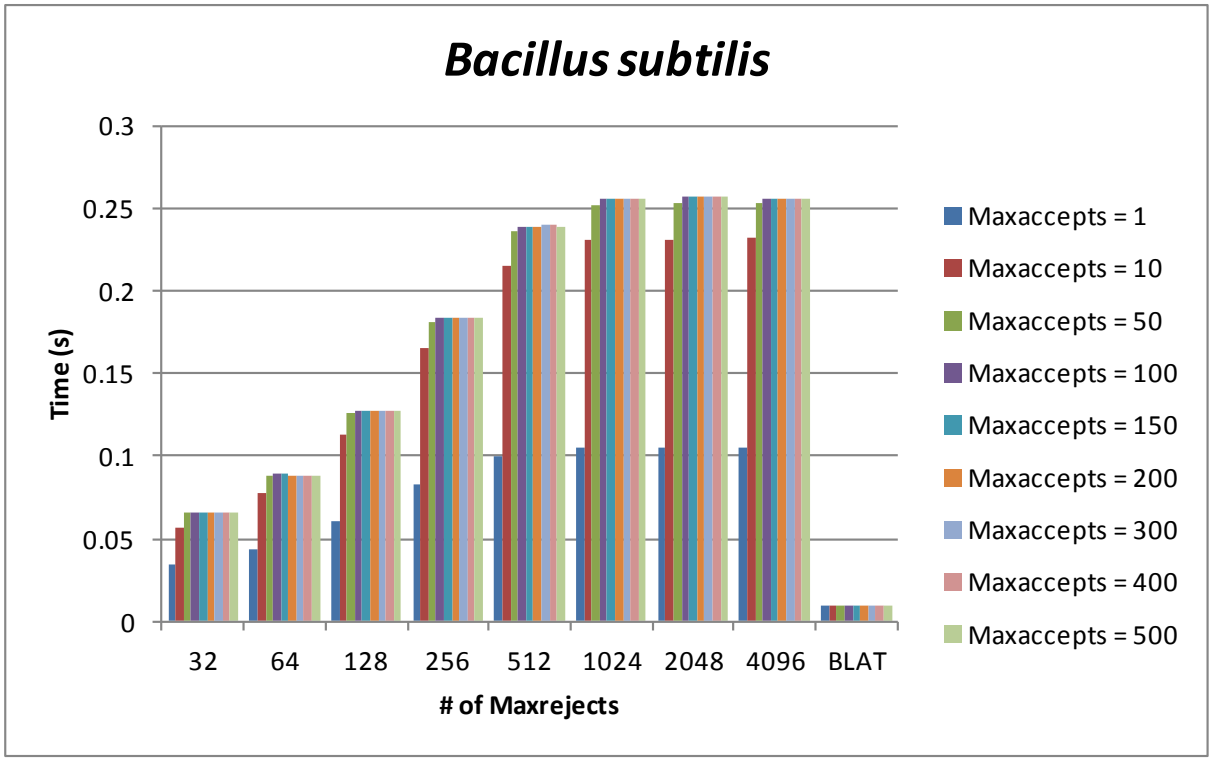


Figure 6- Differences in the execution time required to find homologous sequences when various combinations of –maxaccepts and –maxrejects were specified. The greater the numerical value of the –maxaccepts and –maxrejects, the longer it took to find potential matches, up to a threshold. Error bars are omitted as the numerical results show the same pattern and there would not be a great deviation as a result. The numbers were normalized against the amount of time required by BLASTP to detect homologs.

As shown in Figure 6, increasing the value of the –maxaccepts and –maxrejects options also increased the execution time required by USEARCH to find potential homologous sequences. This makes sense, as it will take the algorithm more time to test a larger number of target sequences. Looking specifically at the –maxaccepts option, it can be observed that when the number of hits allowed before the search stage is stopped

increases, the time required by the algorithm to search the *B. subtilis* genome for matches also rises. However, this result only occurs to a threshold value of 100 –maxaccepts. After this point, it is not relevant how many hits are allowed, the time taken to find them does not fluctuate. The same pattern is seen for the –maxrejects option. Allowing more target sequences to be rejected before search termination gradually increases execution time but only up to a –maxrejects value of 1024. A plateau effect is observed after this point, and the same amount of time is needed to find matches regardless of the –maxaccepts value used. Further, BLAT remains the fastest algorithm when searching for potential homologs.

Figure 7 shows the same general trend with the number of homologs returned as was seen with the time taken to find them. Homolog numbers increase when a larger amount of targets are tested before search termination; however, at a –maxaccepts value of 100 and a –maxrejects value of 1024, a levelling off of generated matches is observed and the same number of homologs are returned to the user regardless of whether or not a greater –maxaccepts or –maxrejects value is used when running USEARCH. This can be attributed to the U-sorting method the algorithm uses to search a sequence database: increasing the number of tested sequences increases the number of potentially similar sequences that will be found; however, the most similar sequence fragments are tested first and these will be counted as hits. Therefore, no further matches will be found as more of the database is searched because all of the hits were found in the first number of sequences examined. Additionally, the values of –maxaccepts and –maxrejects which are responsible for generating a plateau effect in the homolog data are smaller than what is needed to see a plateau in the time data in particular organisms. This contradicts what would be hypothesized, as the greatest number of homologs should be returned at the highest threshold of time (more time taken, means more sequences searched and more putative homologs generated). This could again be attributed to the U-sort method. The database will continue to be searched until either the –maxaccepts or –maxrejects option values are met (i.e. takes more time) but the most homologs will be found earlier in the search as the target sequences are sorted by decreasing similarity to the query.

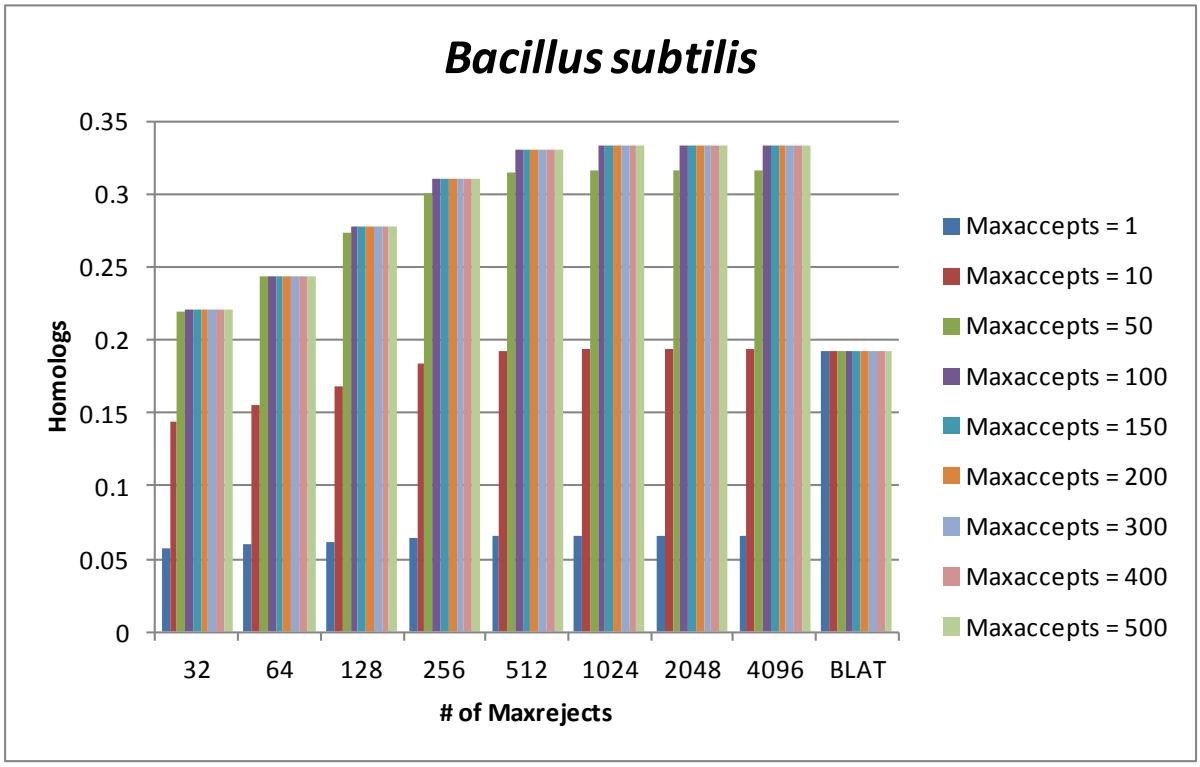


Figure 7- Differences in the number of putative homologous sequences detected by the USEARCH algorithm when various combinations of –maxaccepts and –maxrejects were specified. The greater the numerical value of the –maxaccepts and –maxrejects, the greater the number of matches reported up to a threshold value of 100 for –maxaccepts and 1024 for –maxrejects. Error bars are not shown as the numerical results show the same pattern and there would not be a great deviation as a result. The numbers were normalized against the corresponding number of homologs detected by BLASTP.

Overall, it can be extrapolated that as the number of –maxaccepts increases, there is a parallel increase in the number of matches returned as well as the time taken to find them. This trend is also consistent with rising the –maxrejects option. Further, above a –maxaccepts value of 100, it does not matter how many more –maxaccepts the user specifies, the number of homologs returned and the amount of time taken to find them is the exact same at each corresponding –maxrejects value. Despite returning the same values, a constant pattern is observed where, at and above a –maxaccepts value of 100, a pronounced increase in the amount of data returned grows as the –maxrejects are increased. This only occurs up to a particular threshold as well, at a value of 1024 (in this case). Therefore, it is the –maxaccepts option that dictates the amount of time taken to find matches and how many homologous sequences are detected. After its threshold value has been reached, the responsibility of speed and sensitivity falls on the –maxrejects option (again, only up to a threshold). Also, although BLAT may be faster with respect to

execution time, it fails to locate the number of homologs that are reported by both USEARCH and BLASTP.

3.2.1.2 Ortholog Detection

Following detection of homologous sequences with different USEARCH command lines, RBHs were used to determine how many of the homologs found between *B. subtilis* and *E. coli* were more specifically deemed to be orthologs. The various command lines were run with a PERL program, which facilitated the testing of different combinations of –maxaccepts and –maxrejects in USEARCH. The homolog data generated was subjected to a second PERL program, where the information was searched and pairs of matched proteins with the highest score were extracted using the RBH definition of orthology. From this isolated list, those that appeared to have the greatest degree of match were used to identify orthologs. The number of orthologous pairs detected at each –maxaccepts and –maxrejects combination is shown in Figure 8. The results were normalized to the number of orthologs found by BLASTP.

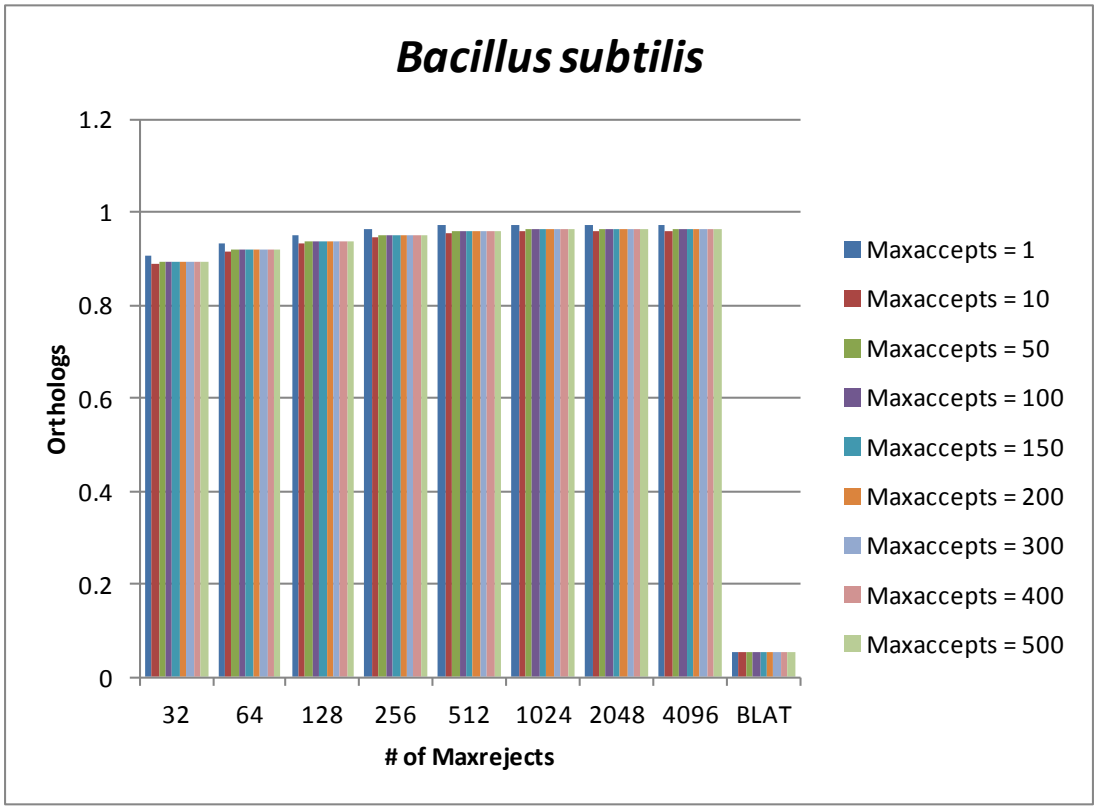


Figure 8- Differences in the number of putative orthologs as RBHs detected by each algorithm. USEARCH and BLASTP detect a similar number of orthologs whereas BLAT detects a much lower number of orthologous genes. Error bars are omitted as the numerical results show the same pattern and there would not be a great deviation as a result. The numbers were normalized against the number of orthologs detected by BLASTP.

It can be seen that the greatest number of orthologs found was at a –maxaccepts value of 1 at each corresponding –maxrejects value. However, it is believed that this result is deceiving as it is not likely that a greater number of orthologs would have been detected when only one hit was allowed before search termination. Further, the number of RBHs detected at this parameter is noticeably different than the number generated at all other higher –maxreject values. The database search method of U-sorting employed by USEARCH may be responsible for this effect. It relies on the fact that the first hit is likely to be the best hit, that is, the sequence with the greatest degree of similarity to the query, but this is not necessarily the case. It is likely then that the chance of finding true orthologs is greater when a –maxaccepts value of 10 and higher is specified, allowing for a more in depth search of the database and therefore giving the algorithm the opportunity to return more accurate hits.

Moreover, the same general trend is seen here as has been observed so far: the number of orthologs returned correlates to the number of target sequences tested as well as the number of homologous sequences found. However, when USEARCH reaches a certain –maxaccepts and –maxrejects combination, in this case values of 50 and 1024, respectively, there is no further increase in detected orthologs. This is due to the search method of U-sorting employed by USEARCH.

Based on the graph, it can also be concluded that although USEARCH is not capable of detecting the number of homologs that BLASTP does, it can find a similar number of orthologs. This may also be due to the ability of USEARCH to sort sequences in order of decreasing number of sequence fragments in common with the query of interest. Comparing sequences in this manner allows USEARCH to efficiently find sequences that have a high degree of similarity. Because orthologs commonly retain their function in their respective organisms, their protein sequence is likely to be conserved and as a result, the USEARCH algorithm will be able to locate them without much difficulty.

One result that is readily noticed, is that BLAT fails to detect anywhere near the number of orthologs of the other two algorithms. This indicates that BLAT's speed negatively affects its sensitivity. Because of this result, BLAT was excluded from further experimentation.

3.2.1.3 Determining Ortholog Quality

When detecting orthologs from a list of homologs, the total number of orthologous genes found is reported; however, as discussed earlier, errors can be made during this search process and so the numbers generated may not be accurate. As a result, to attain a better measure of the true number of orthologs detected, the number of mistakes (genes reported to be orthologs which were not actually orthologous pairs as their adjacency was not conserved) and the number of missing orthologs (no ortholog was found elsewhere in the target upon detection of a paralog) was determined. In this way, the quality and accuracy of the orthologous hits returned by USEARCH in comparison to BLASTP could be estimated.

Error rates were determined through the employment of the conservation of gene order concept which states that if two genes are true orthologs, they will be found adjacent to one another in the target sequence as well as in the corresponding query sequence. There

were two types of mistakes that were searched for; these included the mislabelling of pairs and the failure to detect existing orthologs.

The original ortholog results were run in a PERL program written to compare gene order in the query and target sequences. In so doing, a mislabelling of pairs error was identified when a homologous gene pair was labelled as orthologous when, upon further investigation, one of the genes was a true ortholog while the other was a paralog. If the PERL program was able to find an ortholog to the detected paralog, this further confirmed that a mislabelling mistake had been made. If however, a true ortholog to the query was not found in the target during the search, a missing ortholog error was reported.

The estimated error rate was calculated by dividing the sum of mislabelled and missing orthologs by the number of correct orthologs (that is, the total number of detected orthologs including any potential mistakes). Therefore, the overall error rate is dependent on the number of total errors found as well the total number orthologs detected using each command line. Figure 9 depicts these error rate estimates for each USEARCH option combination as well as for BLASTP. Although a large number of orthologs were detected by USEARCH at a –maxaccepts value of 1, there is a relatively large error rate associated with these results. This confirms the idea that the first match located is not always the one with the greatest sequence similarity to the query and as a result, the ortholog data generated will have a larger number of mistakes. As an extension, one of the highest error rates is found with the default –maxaccepts and –maxrejects combination of 1 and 32, respectively. Although this option requires the least amount of computational time to locate homologs and orthologs, the quality of the results is not good. In contrast, the lowest error rate at each –maxrejects option was associated with a –maxaccepts value of 10. This indicates further that the first hit may not necessarily be the best one; allowing 10 hits before search termination affords the algorithm the ability to exercise greater sensitivity in detecting true orthologs. Further, as the –maxaccepts is increased passed a value of 10, the error rate escalated and plateaued at this elevated level. It may be the case that due to USEARCH's U-sorting method, permitting a larger number of accepts, while generating more data, does not enable the algorithm to find better quality results as those with less error will be found within the first number of sequences tested.

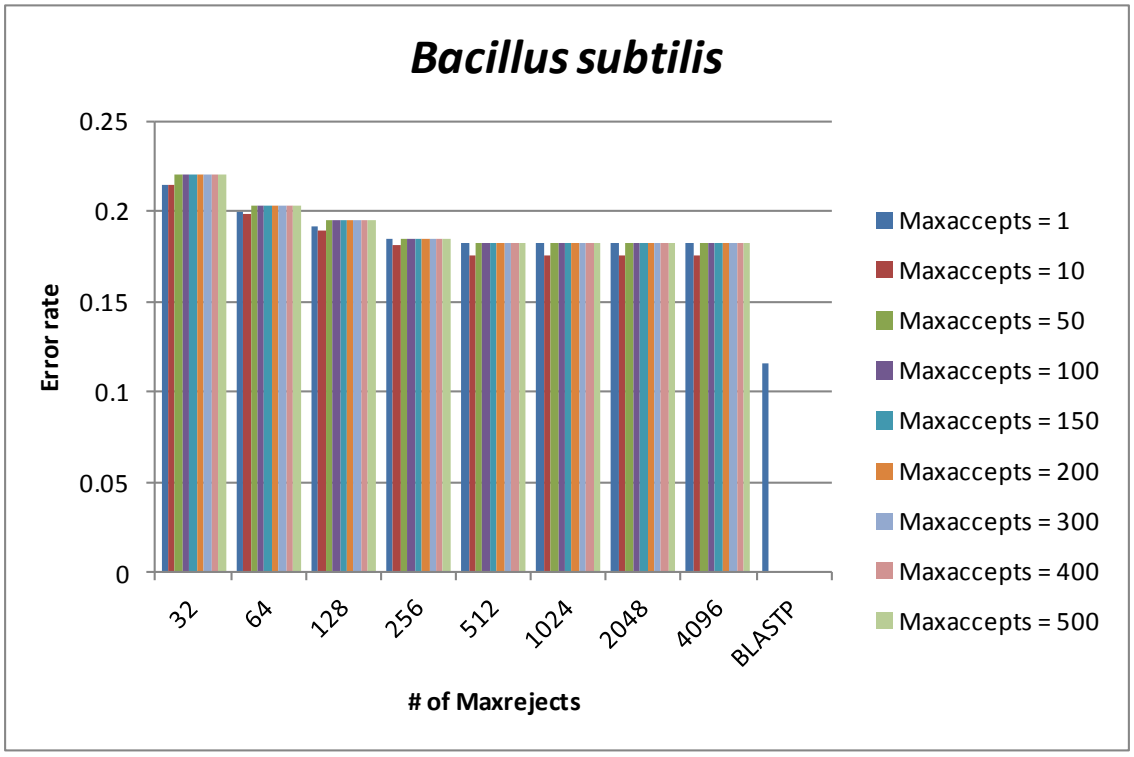


Figure 9- Error rate estimates in ortholog detection by USEARCH in comparison to BLASTP. Estimates were calculated by dividing the mistaken (mislabelled) and missing orthologs by the total number of orthologs detected (adjacent orthologs or an ortholog beside a paralog). Error bars are omitted as the numerical results show the same pattern and there would not be a great deviation as a result.

It can also be concluded that as the number of –maxrejects is increased, the error rate associated with ortholog detection goes down. However, the mistakes made by USEARCH when *B. subtilis* is used as the target, only decrease to a threshold value of 512 rejected sequences; this result complements other results attained thus far. As USEARCH is commanded to test more target sequences prior to search termination, the more sensitive this algorithm becomes. However, in parallel with the –maxaccepts option, the U-sort method will ensure that a large number of target sequences do not need to be rejected before a hit is found and therefore, a threshold is met whereby allowing more mismatches will not have an effect on the number of genuine orthologs found.

Overall, in the specific case of ortholog detection, increasing the sensitivity of the search stage provides the algorithm with a greater chance to return true orthologs and as a result, a lower error rate is observed at each increasing –maxaccepts and –maxrejects combination. Therefore, when using the genome of *B. subtilis* as the target, to obtain ortholog data with a lower associated error rate, a –maxaccepts value of 10 and a

–maxrejects value of 512 could be specified. Finally, the ortholog results given by BLASTP show to be of higher quality than USEARCH at every combination of –maxaccepts and –maxrejects indicated, as there was found to be a lower error rate associated with them. This proves the accuracy of BLASTP.

3.2.2 Nine other genomes as targets

3.2.2.1 Speed and Homolog Detection

In addition to testing *Bacillus*, nine other microorganism genomes were also tested against *E. coli* K12; the results generated when the *Salmonella enterica* genome was used are shown and the remainder of the results are depicted in the Appendix. As illustrated in Figure 10, when *S. enterica* was used as the target genome tested against *E. coli* K12, a similar pattern is observed as was established with the *B. subtilis* target. Generally, it takes USEARCH a longer time to search the target database for hits as the –maxaccepts value is increased; this steady increase in time is observed until a –maxaccepts of 100 is reached. As observed previously, this coincides with the fact that as one indicates a higher –maxaccepts value, more sequences are tested, leading to a greater execution time. Above this value, there are slight variations in the execution time; that is, the plateau effect, as was observed with *B. subtilis*, is not as distinct. This result may not have any significant meaning, as the overall trend of increasing execution time is still observed. Further, at lower –maxrejects, the execution time used by the algorithm actually decreases when –maxaccepts greater than 100 are indicated. From this result, it can be concluded that the number of matches found before the search stage is stopped is responsible for the increase (or subsequent decrease) in time. However, the prevailing trend applies to the –maxrejects option: as the number of –maxrejects specified was raised, the time needed to search the target genome increased in parallel. Allowing more target sequences to be rejected based on dissimilarity to the query forces the algorithm to search more of the target genome to locate hits, and this takes more computational time.

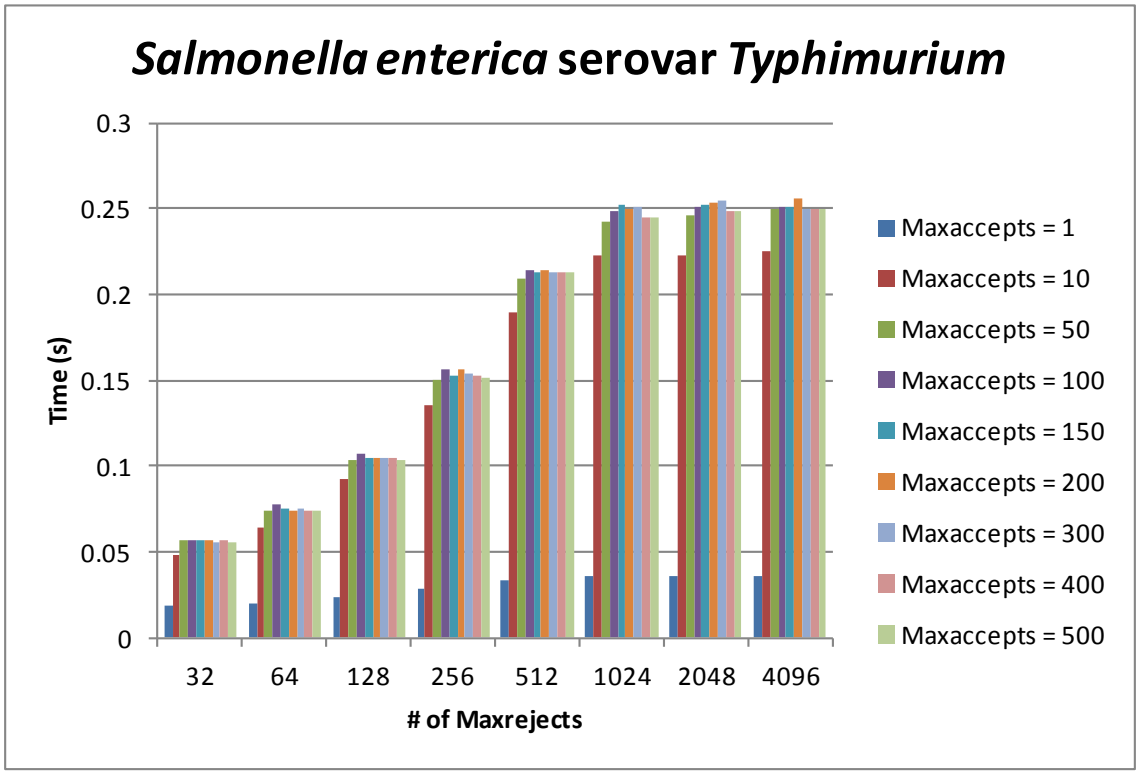


Figure 10- Variations in the amount of time taken by USEARCH to search the genome of S. enterica for matching homologs to the *E. coli* K12 query upon specification of various combinations of –maxaccepts and –maxrejects options. Error bars are not depicted as the numerical results show the same pattern and there would not be a great deviation as a result. The numbers were normalized against the amount of time taken by BLASTP to search the target.

These trends were also observed from the data generated when the genomes of the eight other microorganisms were used as targets to the *E. coli* K12 query (results in Appendix; Figures A3.1.1-A3.1.8). There were variations in the number of –maxaccepts and –maxrejects needed to reach a threshold level of time, that is, depending on the genome under comparison, a different number of matching and dissimilar sequences were tested before execution time reached its maximum. In some instances, this correlates with the evolutionary distance between *E. coli* and the target, where, hypothetically, those genomes that are separated at greater phylogenetic distances will have less sequence similarity. Therefore, a smaller number of sequences will be available as matches (and more as rejections) at a threshold level of time. This was observed with *Methanocaldococcus* and *Saccharomyces* (belonging to archaea and eukarya domains, respectively), where a lower --maxaccepts value of 50 was needed to see a plateau in time. Additionally, more sequences were rejected (therefore, more dissimilarities) from the yeast genome at the threshold level of time.

The number of homologs detected between *S. enterica* and the *E. coli* query rises in parallel with the execution time required of the algorithm as depicted in Figure 11. This is the same result seen when *B. subtilis* was tested. The search method employed by USEARCH is responsible for this trend: as one increases the number of matched target sequences to be tested, the execution time needed to find the matches will also increase. As a result, however, the sensitivity of the algorithm will increase and the number of matches obtained will be thus reflective. Figure 11 gives a very good illustration of the recurring pattern observed throughout the course of this work. Like the results generated when *E.coli* was queried against *B. subtilis*, there is a steady increase in the number of potential homologous sequences found until a combination of 100 –maxaccepts and 1024 –maxrejects was used. The number of results given in the output remain constant for each subsequent –maxaccepts and corresponding –maxrejects option specified. It can also be noted that BLASTP is still able to return more homologs than USEARCH; however, the results returned by these algorithms are comparable. Further, the difference in the number of homologs found by USEARCH and BLASTP was greater when *B. subtilis* was used as the target genome; this is to say that USEARCH detected more homologs when *S. enterica* was the target. This may be due to the fact that *E. coli* and *Salmonella* are closely related evolutionarily speaking and so it would be easier for USEARCH to locate homologous sequences.

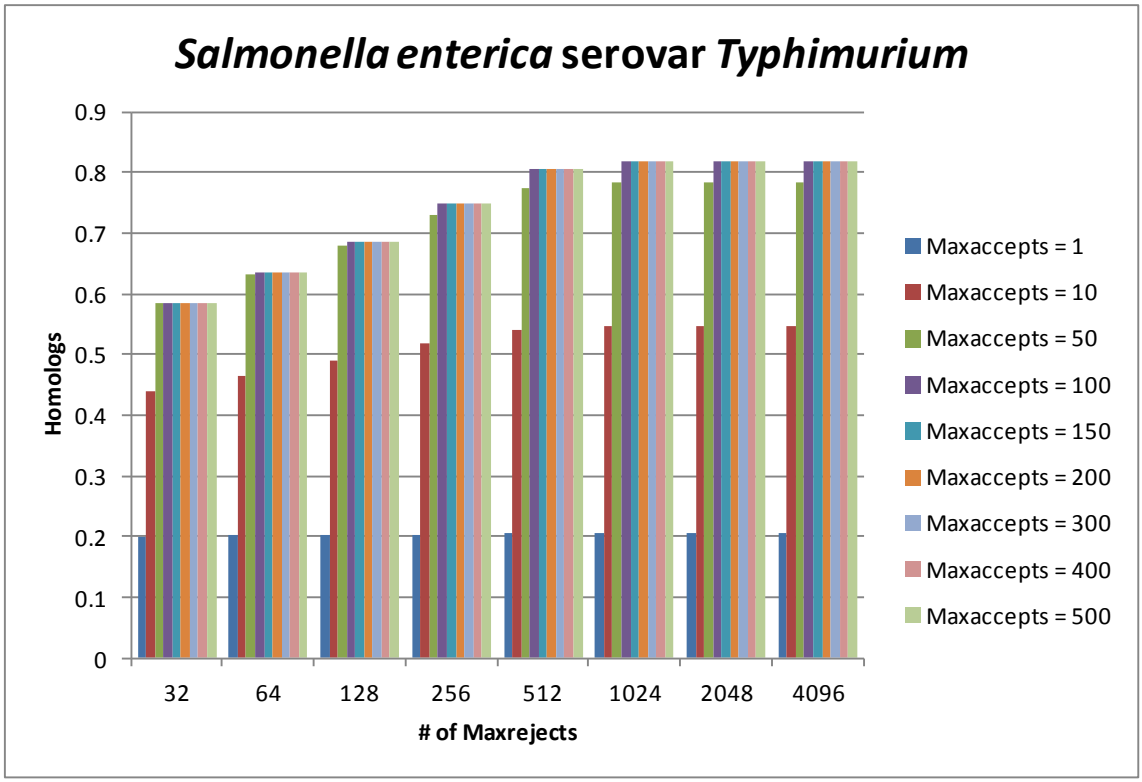


Figure 11- Variations in the number of putative homologs detected by USEARCH when different combinations of –maxaccepts and –maxrejects options were indicated. The genome of *S. enterica* was used as the target to the *E. coli* K12 query. Error bars are not depicted as the numerical results show the same pattern and there would not be a large deviation as a result. The numbers were normalized against the number of homologs returned by BLASTP.

These trends were also observed with the other eight microorganism genomes tested (Appendix Figures A3.2.1-3.2.8), despite their evolutionary distance from *E. coli*. It is of interest to note that, although USEARCH is not as reliable as BLASTP when it comes to homolog detection, the two algorithms do detect a very similar number of homologs in the *Saccharomyces cerevisiae* genome (Figure A3.2.8). This could indicate that as well as being able to detect homologs in closely related species, USEARCH has the ability to find homologous sequences in more divergent organisms.

3.2.2.2 Ortholog Detection

Homologous sequences found with *S. enterica* as the target were then tested to establish how many of those homologs had been derived via a speciation event. As stated previously, USEARCH command lines specifying different –maxaccepts and –maxrejects combinations, were run using a PERL program and the number of orthologs found were normalized to the scores generated by BLASTP. As can be seen in Figure 12, the number of orthologous pairs detected by each –maxaccepts and –maxrejects combination increases steadily along with the option value; this correlates with the number of homologs found. Increasing the number of –maxaccepts also raises the number of orthologs found; however, unlike previous results, the quantity of matches rises in so far as a –maxaccepts value of 10 at each corresponding –maxrejects value. In this instance, the maximum number of orthologs detected occurs at a much lower number of hits, when compared to the results of *B. subtilis.* As stated with homolog detection, this could be due to the closer phylogenetic relationship that *E. coli* and *Salmonella* share in combination with the U-sorting method unique to USEARCH. *E. coli* and *B. subtilis* are not as closely related, and this gives rise to more divergent genetic sequences. Therefore, USEARCH may need to test a larger number of targets in *B. subtilis* before a certain number of hits are located.

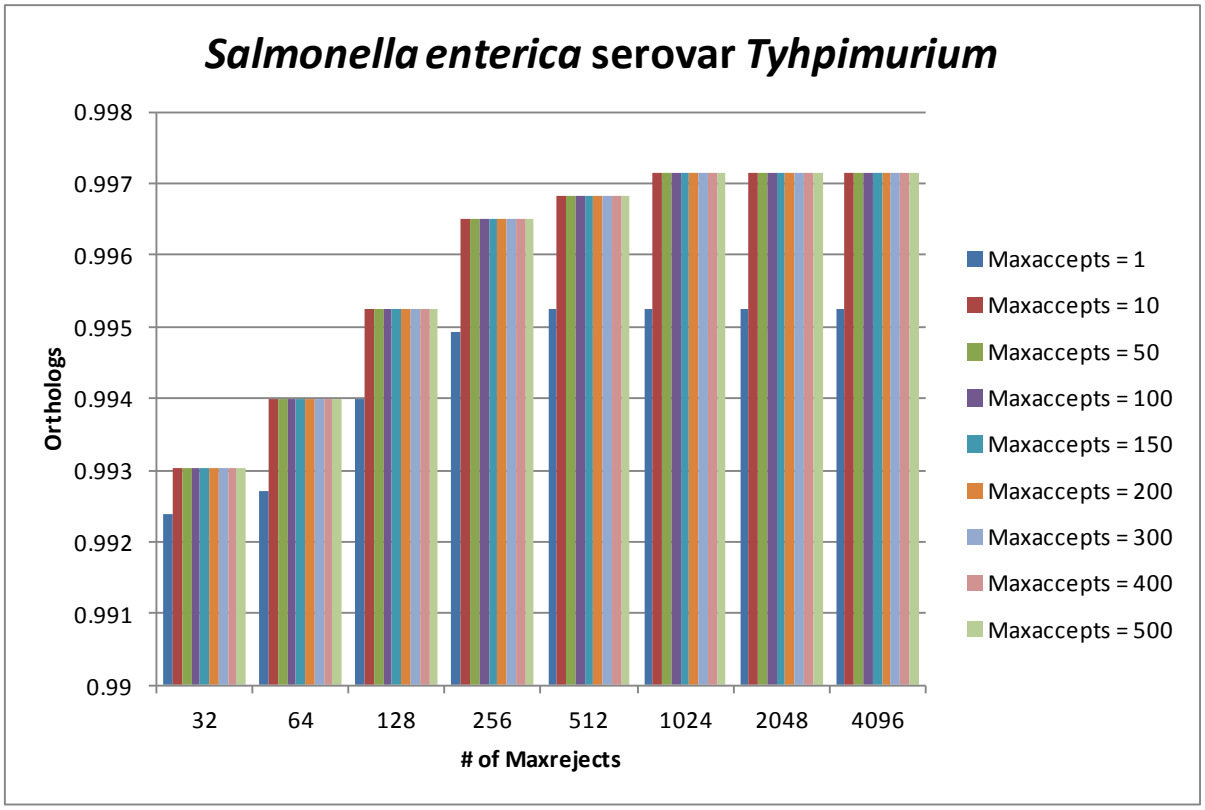


Figure 12- Differences in the number of putative orthologs detected by USEARCH when different combinations of –maxaccepts and –maxrejects options were specified. The genome of *E.coli* K12 was used as the query against *S. enterica*. Error bars are not depicted as the numerical results show the same pattern and there would not be a large deviation as a result. The numbers were normalized against the number of orthologs detected by BLASTP.

It is also noticed that there are a large number of orthologs detected at a –maxaccepts value of 1. This result is similar that found when *B. subtilis* was used as the target and a –maxaccepts of 1 was also indicated. The quality of this data was tested to assess its validity. Additionally, USEARCH is very competent in locating orthologs as it can find almost the same number of orthologous sequences relative to BLASTP.

Again, the trend of increasing ortholog numbers with increasing time and homolog data is observed in the eight other organisms (Figures A3.3.1-A3.3.8). In *Fusobacterium nucleatum*, *Nitrobacter winogradskyi* and *Bordetella pertussis*, the greatest number of orthologs detected is at a –maxaccepts value of 10, regardless of the –maxrejects value specified. This finding could be associated with the relatively close evolutionary relationship these bacteria share to *E. coli*; because there is a greater possibility that they will share sequences that meet a particular similarity threshold, it will be the –maxaccepts option determining the number of orthologs found. A smaller number of sequences will

need to be rejected based on their failure to be matched, and consequently, the number of orthologs detected will not heavily rely on specifying a greater number of –maxrejects.

3.2.2.3 Determining Ortholog Quality

Error rate estimates were calculated with respect to the data generated from ortholog detection. Easily seen in Figure 13 is the fact that there is a high error rate associated with a –maxaccepts value of 1, again proving that the first hit does not always provide the user with the most accurate information. There is less of a general downward trend in calculated mistakes: the same number of errors and as well as correct orthologs is detected at each corresponding –maxaccepts and –maxrejects combination (except for 1 –maxaccepts) above a –maxrejects value of 64. This is likely due to the fact that there is a very small increase in the number of correct orthologs detected at different options and as a result, a smaller variation in the number of errors was computed. Taken as a whole, regardless of the number of target sequences the user specifies to be tested by USEARCH, the exact same error rates will be observed after 64 --maxrejects.

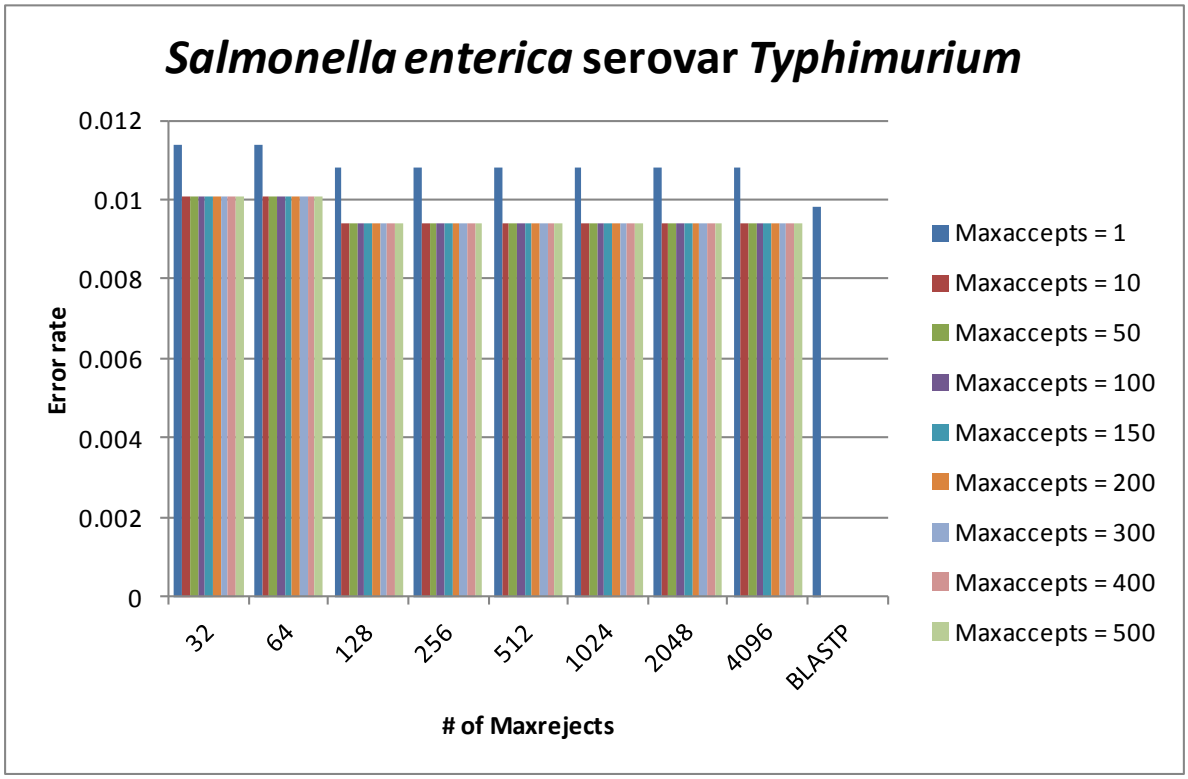


Figure 13- Error rate estimates in ortholog detection by USEARCH in comparison to BLASTP. Estimates were calculated by dividing the mistaken (mislabelled) and missing orthologs by the total number of orthologs detected (adjacent orthologs or an ortholog beside a paralog). Error bars are omitted as the numerical results show the same pattern and there would not be a great deviation as a result.

When comparing the error rates calculated when *B. subtilis* was used as the target to that for *S. enterica*, it can be observed that the overall number of mistakes made by USEARCH is less when *Salmonella* was tested than the number made when *Bacillus* was the target. Even relatively speaking, when compared to ortholog error rates detected with the other eight microorganisms, the number of inaccuracies found with the *S. enterica* genome is fewer. This again may be simply due to the fact that USEARCH is better able to detect true orthologs when the two genomes under comparison are closely related.

It is interesting to note that, in contrast to the BLASTP results generated for every other parameter investigated (except for time), when assessing error rate, BLAST allows a greater number of mistakes in the total number of orthologs found in *S. enterica*, when compared to several USEARCH option combinations. This provides more evidence for the idea that the matched sequences reported by USEARCH when closely related genomes are

under comparison, although not as numerous as BLASTP, show a high degree of similarity to the query and are of good quality.

The finding that the –maxaccepts value of 1 allows for the greatest amount of error when assessing ortholog numbers was maintained in the results for the eight other microorganism genomes. Additionally, in some cases, the overall highest error rate was found at the default –maxaccepts and –maxrejects values. This signifies that simply leaving the default options in place does not afford the user with high quality results; the number of target sequences to be tested needs to be increased so sensitivity can be elevated, thus allowing for better quality results.

The same general trend was observed with these other organisms, where, as the –maxaccepts and –maxrejects are increased, the error rates decrease to a threshold. The point at which the lowest plateau is observed varies, but generally, it is around a –maxaccepts of 50 and a –maxrejects of 512. BLASTP in all instances maintains a lower error rate associated with detected orthologs than does USEARCH.

4 Conclusions

The main purpose of this study was to determine whether the faster speed of a database search algorithm has a profound impact on the sensitivity and quality of results it generates when detecting putative homologous and orthologous sequences and if gaining speed is worth the potential sacrifice in accuracy. At the outset of the investigation, it was found that while BLASTP may be much slower at detecting homologs, its sensitivity was much greater than either USEARCH or BLAT. Although much faster than either USEARCH or BLASTP in execution time, BLAT lacked sensitivity when it was used to detect homolgous and ortholgous sequences between *E. coli* and *B. subtilis*. Therefore, it was concluded here that speed, especially in the case of BLAT, hinders algorithm sensitivity.

However, when testing various combinations of –maxaccepts and –maxrejects options, it was found that increasing the number of target sequences tested can increase the sensitivity of USEARCH (due to an increasing amount of data generated) but does negatively affect the execution time of the algorithm. Further, the number of –maxaccepts

specified determined the execution time as well as the number of homologous and orthologous sequences found but only up to a particular threshold value. For both *B. subtilis* and *S. enterica* targets, this threshold number was a –maxaccepts value of 100 for time and homolog data and was around 50 to 100 --maxaccepts for the eight other microorganisms. Moreover, a threshold –maxaccepts of 50 and 10 were observed for ortholog detection in both *Bacillus* and *Salmonella*, respectively and hovered around values of 50 and 100 –maxaccepts for the other eight genomes. Beyond this, it was the –maxrejects option that influenced the search time and the number of matches in the output; however, it too had a general threshold value of 1024 –maxrejects for all 10 genome targets investigated. Therefore, it can be said that specifying a –maxaccepts value of 100 and a –maxrejects value of 1024 strikes a balance between speed and sensitivity; these options will provide the user with a more sensitive search that generates higher quality results which can still be generated at a faster rate than BLAST. If comparing genomes that are known to be closely related phylogentically, a smaller –maxaccepts value can be specified. Despite increasing USEARCH sensitivity through the manipulation of options however, BLASTP still remained the most sensitive algorithm.

Although USEARCH detected fewer homologs than BLASTP, in terms of RBHs, the two algorithms report a similar number of orthologous sequences when *E. coli* was tested against all 10 genomes. This was the case, however, only when a change in USEARCH options was made. When default options are specified, BLASTP well outperforms USEARCH. Further, the error rate associated with derived putative orthologs decreased as the number of specified –maxaccepts and –maxrejects was increased. This implied that by forcing USEARCH to test a greater number of sequences, there is a greater chance the algorithm will return better quality results. Additionally, the error rate was less using BLASTP in comparison to USEARCH when the genome of *B. subtilis*, as well as the genomes of the other eight microorganisms, was used as the target. This was not the case when *S. enterica* was the target and as more mistakes were actually found when BLASTP was used. This indicates that USEARCH may be quite good at detecting orthologs in closely related species. Moreover, the default option values of 1 –maxaccepts and 32 –maxrejects are not reliable for true ortholog detection as the error rate (and therefore inaccuracy) associated with the number of orthologous sequences found at these values is one of the highest or is the highest when compared to other options tested.

Overall, BLAST appears to be a superior algorithm with respect to sensitivity and output quality. Although it takes more time to locate homologs and orthologs, the amount of data generated and the accuracy of those results is still better than those produced by either USEARCH or BLAT. Speed does hinder sensitivity as well as result quality, although this relationship can be diminished in USEARCH by specifying different combinations of –maxaccepts and –maxrejects.

References

Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ. 1990. Basic Local Alignment Search Tool. *J. Mol. Biol*. 215: 403-410.

Altschul SF, Madden TL, Schaffer AA, Zhang S, Zhang Z, Miller W, Lipman DJ. 1997. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res*. 25: 3389-3402.

Blattner FR, Plunkett G, Bloch CA, Perna NT, Burland V, Riley M, Collado-Vides J, Glasner JD, Rode CK, Mayhew GF, Gregor J, Davis NW, Kirkpatrick HA, Goeden MA, Rose DJ, Mau B, Shao Y. The complete genome sequence of *Escherichia coli* K-12. 1997. *Science*. 277: 1453-1469.

Bork P. 1998. Predicting function: from genes to genomes and back. *J. Mol. Biol.* 283: 707-725.

Bouman S, Churcher C, Badcock K, Brown D, Chillingworth T, Connor R, Dedman K, Devlin L, Gentles S, Hamlin N, Hunt S, Jagels K, Lye G, Moule S, Odell C, Pearson D, Rajandream M, Rice P, Skelton J, Walsh S, Whitehead S, Barrell B. 1997. The nucleotide sequence of *Saccharomyces cerevisiae* chromosome XIII. *Nature*. 387: 90-93.

Bult CJ, White O, Olsen GJ, Zhou L, Fleischmann RD, Sutton GG, Blake JA, FitzGerald LM, Clayton RA, Gocayne JD, Kerlavage AR, Dougherty BA, Tomb JF, Adams MD, Reich CI, Overbeek R, Kirkness EF, Weinstock KG, Merrick JM, Glodek A, Scott JL, Geoghagen NS, Venter JC. 1996. Complete genome sequence of the methanogenic archaeon, *Methanococcus jannaschii*. *Science*. 273: 1058-1073.

Bussey H, Storms RK, Ahmed A, Albermann K, Allen E, Ansorge W, Araujo R, Aparicio A, barrel B, Badcock K, Benes V, Botstein D, Bouman S, Bruckner M, Carpenter J, Cherry JM, Chung E, Churcher C, Coster F, Davis K et al. 1997. The nucleotide sequence of *Saccharomyces cerevisiae* chromosome XVI. *Nature*. 387: 103-105.

Bussey H, Kaback DB, Zhong WW, Vo DT, Clark MW, Fortin N, Hall J, Ouellette BFF, Keng T, Barton AB, Su Y, Davies J, Storms RK. 1995. The nucleotide sequence of chromosome I from *Saccharomyces cerevisiae*. *Proc. Natl. Acad. Sci. USA*. 92: 3809-3813.

Churcher C, Bouman S, Badcock K, Bankier A, Brown D, Chillingworth T, Connor R, Devlin K, Gentles S, Hamlin N, Harris D, Horsnell T, Hunt S, Jones M, Lye G, Moule S, Odell C, Pearson D, Rajandream M, Rice P, Rowley N, Skelton J, Smith V, Walsh S, Whitehead S, Barrell B, 1997. The nucleotide sequence of *Saccharomyces cerevisiae* chromosome IX. *Nature*. 387: 84-87.

Dietrich FS, Mulligan J, Hennessy K, Yelton MA, Allen E, Araujo R, Aviles E, Berno A, Brennan T, Carpenter J, Chen E, Cherry JM, Chung E, Duncan M, Guzman E, Hartzell G, Hunicke-Smith S, Hyman RW, Kayser A, Komp C et al. 1997. The nucleotide sequence of *Saccharomyces cerevisiae* chromosome V. *Nature*. 387: 78-81.

Dujon B, Albermann L, Aldea M, Alexandraki D, Ansorge W, Arino J, Benes V, Bohn C, Bolotin-Fukuhara M, Bordonne R, Boyer J, Camasses A, Casamayor A, Casas C, Cheret G, Czieplucl C, Daignan-Fornier B, Dang DV, de Haan M, Delius H et al. 1997. The nucleotide sequence of *Saccharomyces cerevisiae* chromosome XV. *Nature*. 387: 98-102.

Dujon B, Alexandraki D, Andre B, Ansorge W, Baladron V, Ballesta JPG, Banrevi A, Bolte PA, Bolotin-Fukuhara M, Bossier P, Bou G, Boyer J, Bultrago MJ, Cheret G, Colleaux L, Daignan-Fornier B, del Rey F, Dion C, Domdey H Dusterhoft et al. 1994. Complete DNA sequence of yeast chromosome XI. *Nature*. 369: 371-378.

Eddy SR. 2004. What is dynamic programming? *Nature Biotechnol*. 22: 909-910.

Edgar RC. 2010. Search and clustering orders of magnitude faster than BLAST. *Bioinformatics*. 26: 2460-2461.

Feldmann H, Aigle M, Aljinovic C, Andre B, Baclet MC, Barthe C, Bayr A, Becam AM, Biteau N, Boles E, Brandt T, Brendel M, Bruckner M, Bussereau F, Christiansen C, Contreras R, Crouzet M, Czieplucl C, Demolis N, Delaveau T et al. 1994. Complete DNA sequence of yeast chromosome II. *EMBO J.* 13: 5795-5809.

Fitch WM. 2002. Homology a personal view on some of the problems. *Trends Genet*., 16: 227-231.

Fitch WM. 1970. Distinguishing homologous from analogous proteins. *Syst. Zool.* 19: 99-106.

Fraser CM, Norris SJ, Weinstock GM, White O, Sutton GG, Dodson R, Gwinn M, Hickey ER, Clayton R, Ketchum KA, Sodergren E, Hardham JM, McLeod MP, Salzberg S, Peterson J, Khalak H, Richardson D, Howell JK, Chidambaram M, Utterback T, McDonald L, Artiach P, Bowman C, Cotton MD, Fujii C, Garland S, Hatch B, Horst K, Roberts L, Sandusky M, Weidman J, Smith HD, Venter JC. 1998. Complete genome sequence of *Treponema pallidum*, the syphilis spirochete. *Science*. 281: 375-388.

Galibert F, Alexandraki D, Baur A, Boles E, Chalwatzis N, Chuat JC, Coster F, Czieplucl C, De Haan M, Domday H, Durand P, Entian KD, Gatius M, Goffeau A, Grivell LA, Hennemann A, Herbert CJ, Heumann K, Hilger F, Hollenberg CP et al. 1996. Complete nucleotide sequence of *Saccharomyces cerevisiae* chromosome X. *EMBO J*. 15: 2031-2049.

Jacq C, Alt-Morbe J, Andre B, Arnold W, Bahr A, Ballesta JPG, Bargues M, Baron C, Becker A, Biteau N, Blocker H, Blugeon C, Boskovic J, Brandt P, Bruckner M, Buitrago MJ, Coster F, Delaveau T, del Rey F, Dujon B et al. 1997. The nucleotide sequence of *Saccharomyces cerevisiae* chromosome IV. *Nature.* 387: 75-78.

Janga SC and Moreno-Hagelsieb G. 2004. Conservation of adjacency as evidence of paralogous operons. *Nucleic Acids. Res*. 32: 5392-5397.

Johnston M, Hillier L, Riles L, Albermann K, Andre B, Ansorge W, Benes V, Bruckner M, Delius H, Dubois E, Dusterhoft A, Entian KD, Floeth M, Goffeau A, Hebling U, Heumann K, Heuss-Neitzel D, Hilbert H, Hilger F, Kleine K et al. 1997. The nucleotide sequence of *Saccharomyces cerevisiae* chromosome XII. *Nature*. 387: 87-90.

Johnston M, Andrews S, Brinkman R, Cooper S, Ding H, Dover J, Du Z, Favello A, Fulton L, Gattung S, Geisel C, Kirsten J, Kucaba T, Hillier L, Jier M, Johnston L, Langston Y, Latreille p, Macri C, Mardis E et al. 1994. Complete nucleotide sequence of *Saccharomyces cerevisiae* chromosome VIII. *Science.* 265: 2077-2082.

Kapatral V, Anderson I, Ivanova N, Reznik G, Los T, Kykidis A, Bhattacharyya A, Bartman A, gardner W, Grechkin G, Zhu L, Vasieva O, Chu L, Kogan Y, Chaga O, Goltsman E, Bernal A, Larsen N, D'Souza M, Walunas T, Pusch G, Haselkorn R, Fonstein M, Kyrpides N, Overbeek R. 2002. *Genome sequence and analysis of the oral bacterium Fusobacterium* nucleatum strain ATCC 25586. *J. Bacteriol*. 184:2005-2018.

Kent WJ. 2002. BLAT-The BLAST-Like Alignment Tool. *Genome Res*. 12: 656-664.

Koonin E. 2005. Orthologs, paralogs, and evolutionary genomics. *Annu. Rev. Genet*. 39: 309-338.

Kristensen DM, Wolf YI, Myshegian AR, Koonin EV. 2011. Computational methods for gene orthology inference. *Brief. Bioinform*. 12: 379-391.

Kunst F, Ogasawara N, Moszer I, Albertini AM, Alloni G, Azevedo V, Bertero MG, Bessieres P, Borchert S, Borriss R, Boursier L, Brans A, Braun M, Brignell SC, Brori S, Brouillet S, Bruschi CV, Caldwell B, Capuano V, Carter NM et al. 1997. The complete genome sequence of the Gram-positive bacterium *Bacillus subtilis*. *Nature*. 390: 249-256.

Luca S, Copeland A, lapidus A, Cheng JF, Goodwin L, Pitluck S, Davenport K, Detter JC, Han C, Tapia R, Land M, Hauser L, Kyrpides N, Ivanova N, Ovchinnikova G, Pagani I, Brumm P, Mead D, Woyke T. 2011. Complete sequence of chromosome of *Streptomyces flavogriseus* ATC 33331. Unpublished. Department of Energy Joint Genome Institute.

McBride MJ, Xie G, Martens EC, Lapidus A, Henrissat B, Rhodes RG, Goltsman E, Wang W, Xu J, Hunnicutt DW, Staroscik AM, Hoover TR, Cheng YQ, Stein JL. 2009. Novel features of the polysaccharide-digesting gliding bacterium *Flavobacterium johnsoniae* as revealed by genome sequence analysis. *Appl. Environ. Microbiol*. 75: 6864-6875.

McClelland M, Sanderson KE, Speith J, Clifton SW, Latreille P, Courtney L, Porwollik S, Ali J, Dante M, Du F, Hou S, Layman D, Leonard S, Nguyen C, Scott K, Holmes A, Grewal N, Mulvaney E, Ryan E, Sun H, Florea L, Miller W, Stoneking T, Nhan M, Waterston R, Wilson RK. 2001. Complete genome sequence of *Salmonella enterica* serovar *Typhimurium* LT2. *Nature*. 413: 852-856.

Moreno-Hagelsieb G and Latimer K. 2008. Choosing BLAST options for better detection of orthologs as reciprocal best hits. *Bioinformatics*. 24: 319-324.

Moreno-Hagelsieb G, Trevino V, Perez-Rueda E, Smith TF, Collado-Vides J. 2001. Transcription unit conservation in the three domains of life: a perspective from *Escherichia coli. Trends Genet*. 17: 175-177.

Murakami Y, naitou M, Hagiwara H, Shibata T, Ozawa M, Sasanuma S, Sasanuma M, Tsuchiya Y, Soeda E, Yokoyama K, Yamazaki M, Tashiro H, Eki T. 1995. Analysis of the nucleotide sequence of chromosome VI from *Saccharomyces cerevisiae*. *Nat. Genet*. 10: 261-268.

Needleman SB and Wunsch CD. 1970. A general method applicable to search for similarities in amino acid sequence of two proteins. *J Mol. Biol*. 48: 443-453.

Oliver SG, van der Aart JM, Agostoni-Carbone ML, Aigie M, Alberghina L, Alexandraki D, Antoine G, Anwar R, Ballesta JPG, Benit P, Berben G, Bergantino E, Biteau N, Bolle PA, Bolotin-Fukuhara M, Brown A, Brown AJP, Buhler JM, Carcano C, Carignani G et al. 1992. The complete DNA sequence of yeast chromosome III. *Nature*. 357: 38-46.

Parkhill J, Sebaihia M, Preston A, Murphy LD, Thomson N, Harris DE, Holden MTG, Churcher CM, Bentley SD, Mungall KL, Cerdeno-Tarraga AM, Temple L, James K, Harris B, Quail MA, Achtman M, Atkin R, Baker S, Basham D, Bason N et al. 2003. Comparative analysis of the genome sequences of *Bordetella pertussis*, *Bordetella parapertussis* and *Bordetella bronchiseptica*. *Nat. Genet*. 35: 32-40.

Pearson WR and Lipman DJ. 1988. Improved tools for biological sequence comparison. *Proc. Natl. Acad. Sci. USA*. 85: 2444-2448.

Philippsen P, Kleine K, Pohlmann R, Dusterhoft A, Hamberg K, Hegemann JH, Obermaier B, Urrestarazu LA, Aert R, Albermann K, Altmanss R, Becam AM, Beinhauer J, Boskovic J, Buitrago MJ, Bussereau F, Coster F, Crouzet M et al. 1997. The nucleotide sequence of *Saccharomyces cerevisiae* chromosome XVI and its evolutionary implications. *Nature*. 387: 93-98.

Starkenburg SR, Chain PS, Sayavedra-Soto LA, Hauser L, Land ML, Larimer FW, Malfatti SA, Klotz MG, Bottomley PJ, Arp DJ, Hickey WJ. 2006. Genome sequence of the chemolithoautotrophic nitrite-oxidizing bacterium *Nitrobacter winogradskyi* Nb-255. *Appl. Environ. Microbiol.* 72: 2050-2063.

Tettelin H, Agostoni-carbone ML, Albermann K, Albers M, Arroyo J, Backes U, Barreiros T, Bertani I, Bjourson AJ, Bruckner M, Bruschi CV, Carrignani G, Castagnoli L, Cerdan E, Clemente ML, Coblenz A, Coglievina M, Coissac E, Defoor E, Del Bino S et al. 1997. The nucleotide sequence of *Saccharomyces cerevisiae* chromosome VII. *Nature*. 387: 81-84.

The NCBI Taxonomy Database. [http://www.ncbi.nlm.nih.gov/taxonomy]

Wall DP, Fraser HB, Hirsch AE. 2003. Detecting putative orthologs. *Bioinformatics*. 13: 1710-1711.

Zvelebil M and Baum JO. 2008. Understanding Bioinformatics. Garland Science, Taylor & Francis Group, LLC, NY.

Appendix

A1 An example PERL program

```
#!/usr/bin/perl
### this program will run the Usearch combinations for E coli
K12 MG1655

### against every other genome


##use organize_data; ### library to figure out paths

use Update_lists::DOMAINS; ### get list of genomes


my $query_test = "E_coli_K12_MG1655";

my $subject_test = "B_subtilis_168";


@maxaccepts = qw(
                    1
                    10
                    50
                    100
                    150
                    200
                    300
                    400
                    500
            );
```

```
@maxrejects = qw(
                    32
                    64
                    128
                    256
                    512
                    1024
                    2048
                    4096
            );


my $usearch_command = "usearch -evalue 1e-6 -query
$query_test.faa "
    . "-db $subject_test.faa \\\n"
    . "-userfields "
    .
"query+target+bits+raw+evalue+qs+ts+id4+cols+qlo+qhi+tlo+thi+
ql+tl";


### directories for uncompressed genomes and for results
$genome_dir = "Genomes";
mkdir("$genome_dir") unless( -d "$genome_dir" );


for my $maxaccepts ( @maxaccepts ) {
    for my $maxrejects ( @maxrejects ) {
        my $out_file
            =
"$query_test.$subject_test.$maxaccepts.$maxrejects.usearch";
```

```
        my $full_command = "$usearch_command \\\n"
            . "-maxaccepts $maxaccepts -maxrejects
$maxrejects \\\n"
                . "-userout $out_file";
        print"running:\n time ",$full_command,"\n";
        #system "time $full_command";
    }
}
```

A2 Example command lines

Table A2.1- Command lines used for each of the three algorithms tested. The user can manipulate the command lines to specify desired parameters.

Search Algorithm	Command Line Type	Command Line
USEARCH	Default	usearch –query E_coli_K12_MG1655.faa –db B_subtilis_168.faa -evalue 1e-6 –userout results.USEARCH1 –userfields query+target+bits+raw+evalue+qs+ts+id4+cols+ qlo+qhi+tlo+thi+ql+tl
	Maxaccepts and Maxrejects specified	usearch –query E_coli_K12_MG1655.faa –db B_subtilis_168.faa –maxaccepts 100 –maxrejects 256 -evalue 1e-6 –userout results.USEARCH2 –userfields query+target+bits+raw+evalue+qs+ts+id4+cols+ qlo+qhi+tlo+thi+ql+tl
BLAT	Default	blat –prot B_subtilis_168.faa E_coli_K12_MG1655.faa –out=blast9 output.BLAT
BLAST	Default	blastp –db B_subtilis_168 –query E_coli_K12_MG1655.faa –dbsize 500000000 –evalue 1e-6 –show_gis –num_alignments 4175 –use_sw_tback –outfmt ‘7 qseqid sseqid bitscore score evalue indent positive length qstart qend qlen sstart send slen’ > E_coli_K12_MG1655.B_subtilis_168.blastp

A3 Figures-Eight Other Organisms

A3.1 Testing USEARCH Options: Execution time

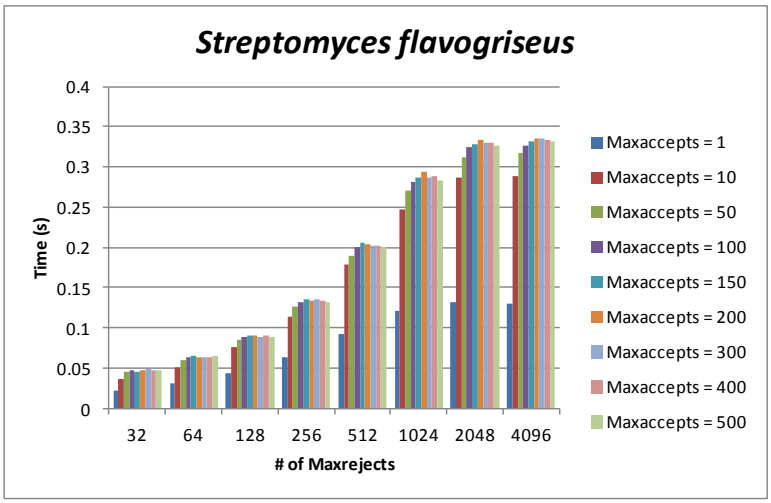


Figure A3.1.1: Execution time taken by USEARCH to detect putative homologs between *E. coli* K12 and *Streptomyces flavogriseus*. Time was normalized against the amount of time taken by BLASTP. As the number of target sequences tested increases, so does the execution time, up to a threshold combination of approximately 150 –maxaccepts and 2048 –maxrejects. USEARCH takes much less time than BLASTP despite increasing option values. Error bars are not depicted as the numerical results show the same pattern and there would not be a great deviation from the mean as a result.

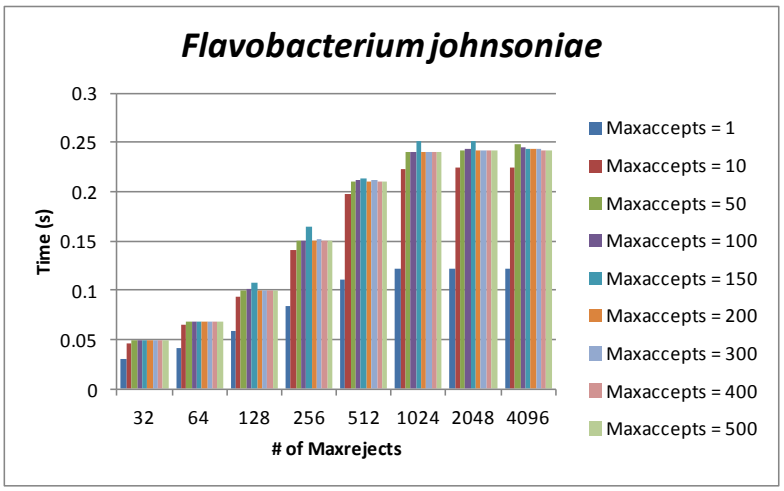


Figure A3.1.2: Execution time taken by USEARCH to detect putative homologs between *E. coli* K12 and *Flavobacterium johnsoniae*. Time was normalized against the amount of time taken by BLASTP. As the number of target sequences tested increases, so does the execution time, up to a threshold combination of approximately 100 –maxaccepts and 1024 –maxrejects. USEARCH takes much less time than BLASTP despite increasing option values. Error bars are not depicted as the

numerical results show the same pattern and there would not be a great deviation from the mean as a result.

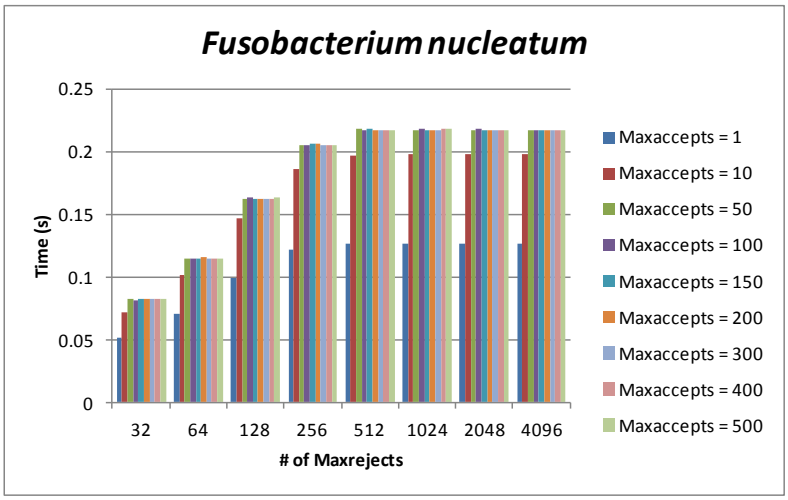


Figure A3.1.3: Execution time taken by USEARCH to detect putative homologs between *E. coli* K12 and *Fusobacterium nucleatum*. Time was normalized against the amount of time taken by BLASTP. As the number of target sequences tested increases, so does the execution time, up to a threshold combination of approximately 50 –maxaccepts and 512 –maxrejects. USEARCH takes much less time than BLASTP despite increasing option values. Error bars are not depicted as the numerical results show the same pattern and there would not be a great deviation from the mean as a result.

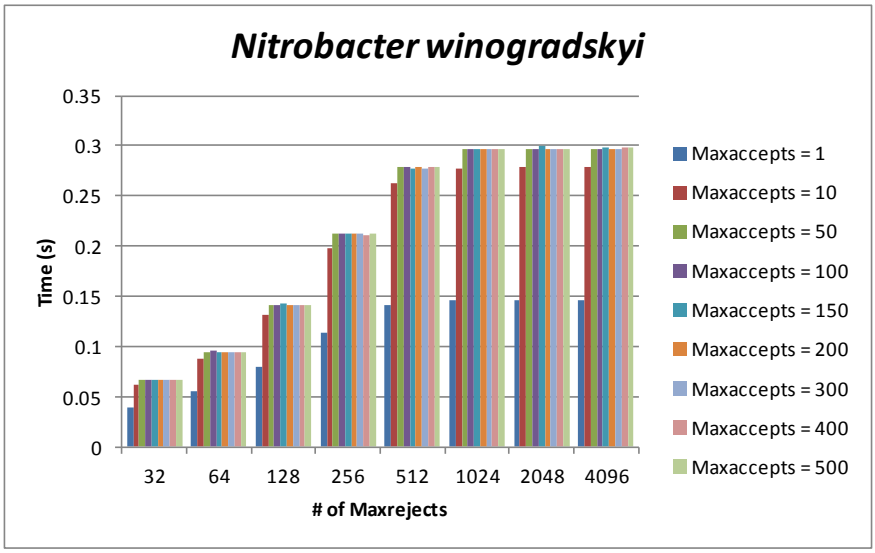


Figure A3.1.4: Execution time taken by USEARCH to detect putative homologs between the *E. coli* K12 and *Nitrobacter winogradskyi*. Time was normalized against the amount of time taken by BLASTP. As the number of target sequences tested increases, so does the execution time, up to a threshold combination of approximately 50 –maxaccepts and 1024 –maxrejects. USEARCH takes much less time than BLASTP despite increasing option values. Error bars are not depicted as the numerical results show the same pattern and there would not be a great deviation from the mean as a result.

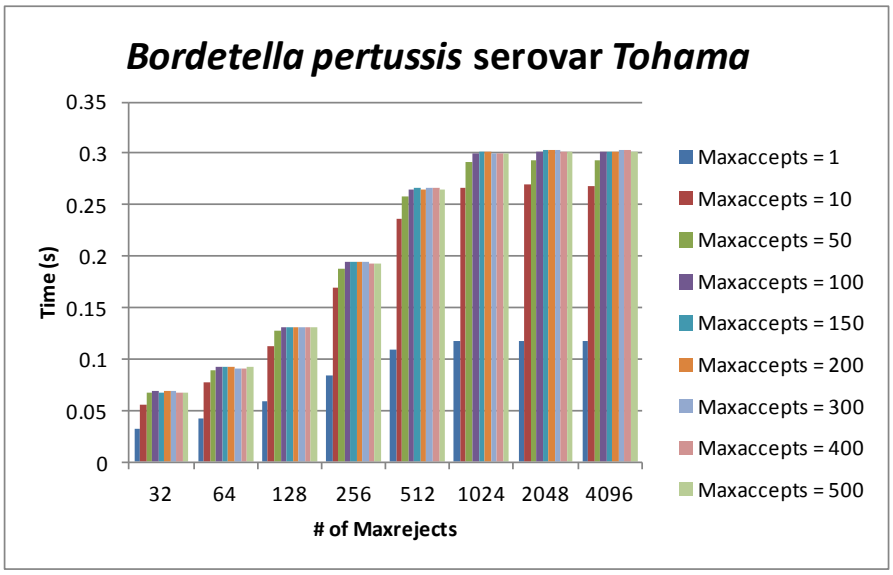


Figure A3.1.5: Execution time taken by USEARCH to detect putative homologs between *E. coli* K12 and *Bordetella pertussis*. Time was normalized against the amount of time taken by BLASTP. As the number of target sequences tested increases, so does the execution time, up to a threshold combination of approximately 100 –maxaccepts and 1024 –maxrejects. USEARCH takes much less time than BLASTP despite increasing option values. Error bars are not depicted as the numerical results show the same pattern and there would not be a great deviation from the mean as a result.

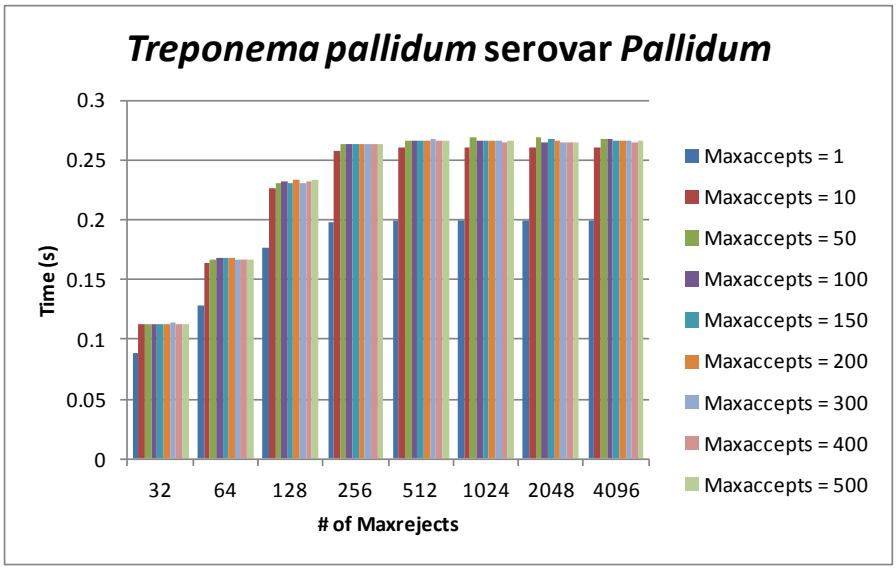


Figure A3.1.6: Execution time taken by USEARCH to detect putative homologs between *E. coli* K12 and *Treponema pallidum*. Time was normalized against the amount of time taken by BLASTP. As the number of target sequences tested increases, so does the execution time, up to a threshold combination of approximately 50 –maxaccepts and 256 –maxrejects. USEARCH takes much less time than BLASTP despite increasing option values. Error bars are not depicted as the numerical results show the same pattern and there would not be a great deviation from the mean as a result.

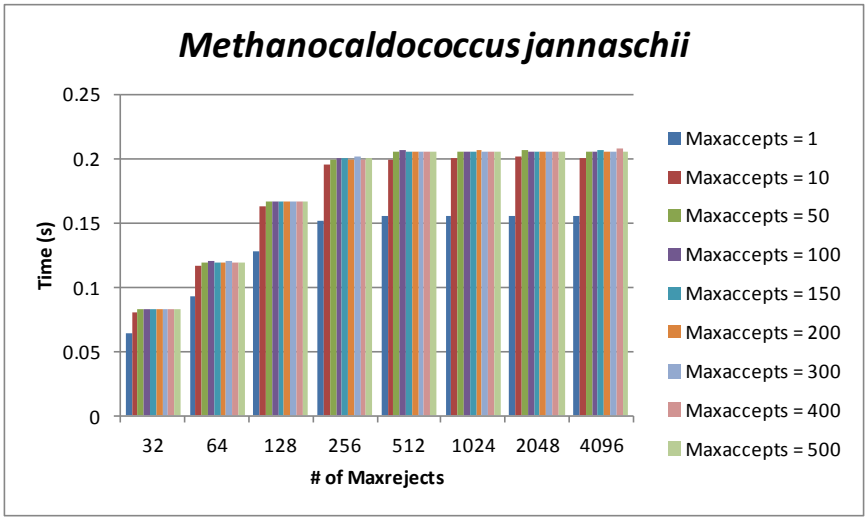


Figure A3.1.7: Execution time taken by USEARCH to detect homologs between *E. coli* K12 and *Methanocaldococcus jannaschii*. Time was normalized against the amount of time taken by BLASTP. As the number of target sequences tested increases, so does the execution time, up to a threshold combination of approximately 50 –maxaccepts and 512 –maxrejects. USEARCH takes much less time than BLASTP despite increasing option values. Error bars are not depicted as the numerical results show the same pattern and there would not be a great deviation from the mean as a result.

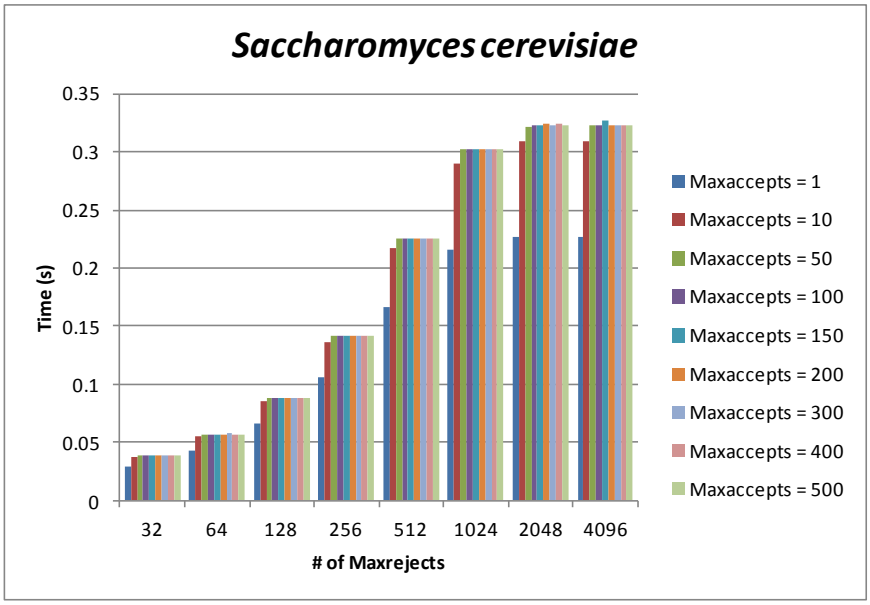


Figure A3.1.8: Execution time taken by USEARCH to detect putative homologs between *E. coli* K12 and *Saccharomyces cerevisiae*. Time was normalized against the amount of time taken by BLASTP. As the number of target sequences tested increases, so does the execution time, up to a threshold combination of approximately 50 –maxaccepts and 2048 –maxrejects. USEARCH takes much less time than BLASTP despite increasing option values. Error bars are not depicted as the numerical results show the same pattern and there would not be a great deviation from the mean as a result.

A3.2 Testing USEARCH Options: Homolog Detection

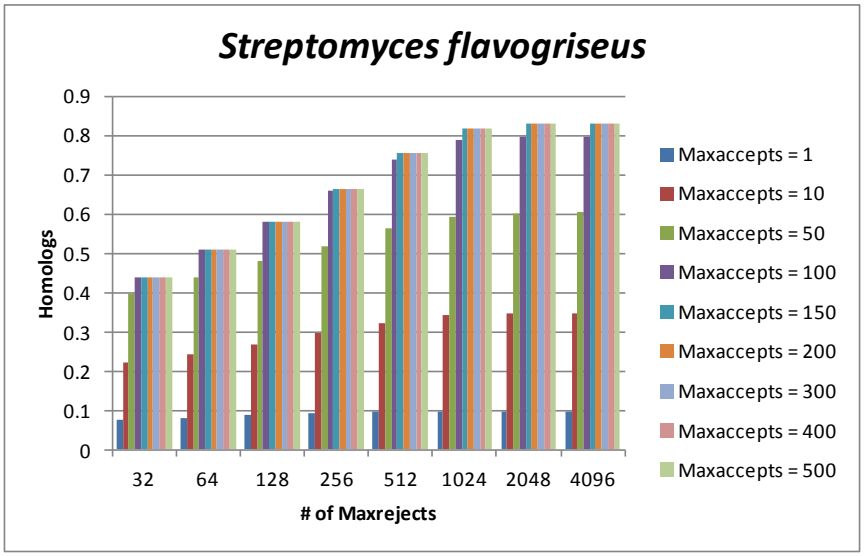


Figure A3.2.1: Number of putative homologs detected by USEARCH when different values of the –maxaccepts and –maxrejects options were specified; the genome of *E. coli* K12 was used as the query and the genome of *Streptomyces flavogriseus* was used as the target. The number of matches increases as the number of target sequences tested increases but only up to a threshold of 150 –maxaccepts and 2048 –maxrejects. This is the same threshold than what is seen for this organism when time was monitored. Results were normalized against the number of homologs detected by BLASTP. Error bars are not depicted as the numerical results show the same pattern and there would not be a great deviation from the mean as a result.

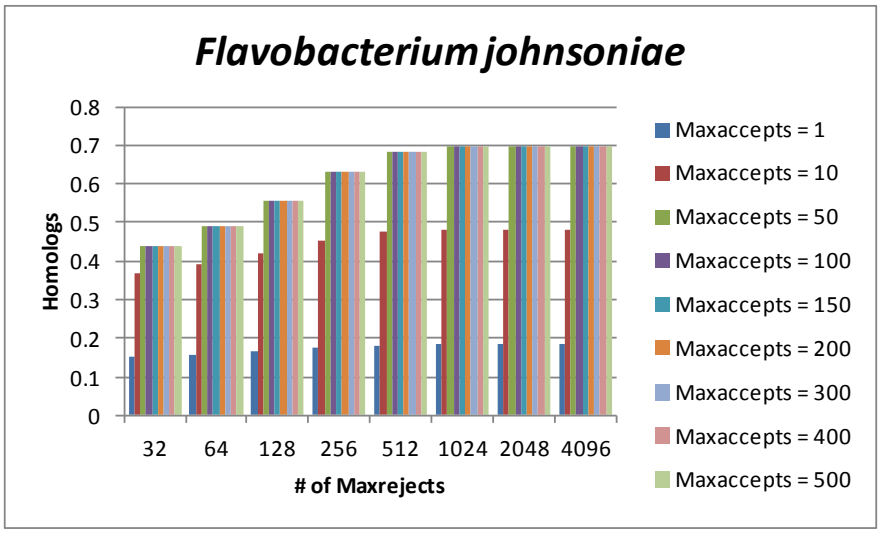


Figure A3.2.2: Number of putative homologs detected by USEARCH when different values of the –maxaccepts and –maxrejects options were specified. The genome of *E. coli* K12 was used as the query and the genome of *Flavobacterium johnsoniae* was used as the target. The number of matches increases as the number of target sequences tested increases but only up to a threshold of 50 –maxaccepts and 1024 –maxrejects. This is a slightly different threshold than what it seen for this organism when time was monitored. Results were normalized against the number of homologs

detected by BLASTP. Error bars are not depicted as the numerical results show the same pattern and there would not be a great deviation from the mean as a result.

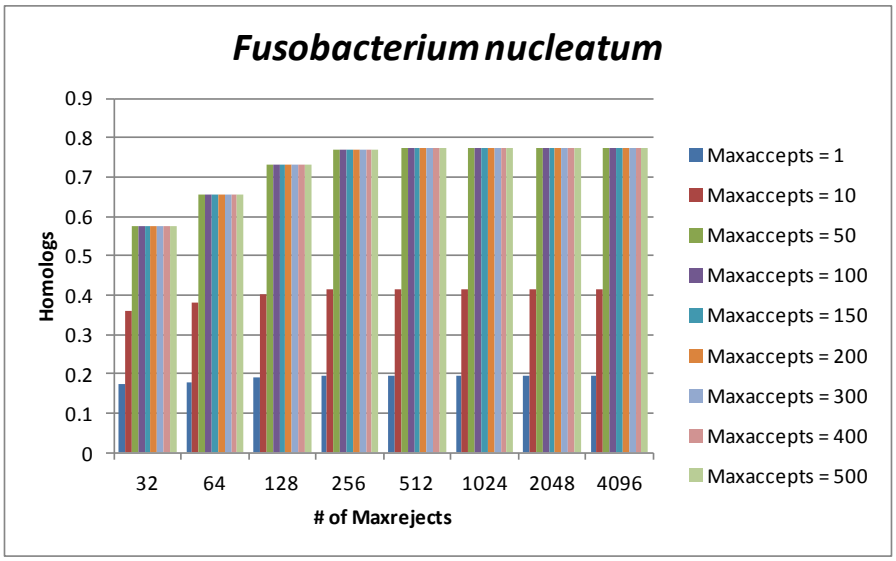


Figure A3.2.3: Number of putative homologs detected by USEARCH when different values of the –maxaccepts and –maxrejects options were specified; the genome of *E. coli* K12 was used as the query and the genome of *Fusobacterium nucleatum* was used as the target. The number of matches increases as the number of target sequences tested increases but only up to a threshold of 50 –maxaccepts and 256 –maxrejects. This is a slightly different threshold than what is seen for this organism when time was monitored. Results were normalized against the number of homologs detected by BLASTP. Error bars are not depicted as the numerical results show the same pattern and there would not be a great deviation from the mean as a result.

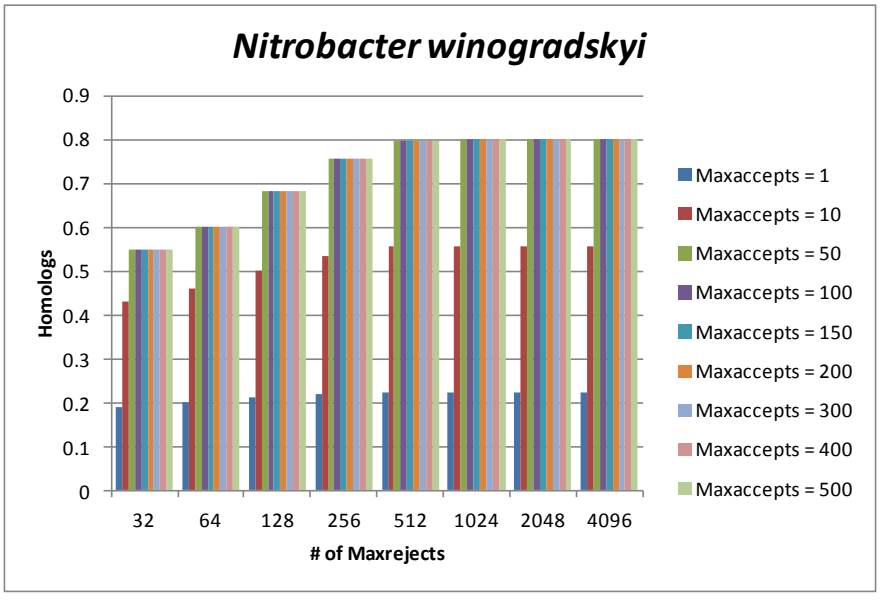


Figure A3.2.4: Number of putative homologs detected by USEARCH when different values of the –maxaccepts and –maxrejects options were specified; the genome of *E. coli* K12 was used as the query and the genome of *Nitrobacter winogradskyi* was used as the target. The number of matches increases as the number of target sequences tested increases but only up to a threshold of 50 –maxaccepts and 512 –maxrejects. This is a slightly different threshold than what is seen for this

organism when time was monitored. Results were normalized against the number of homologs detected by BLASTP. Error bars are not depicted as the numerical results show the same pattern and there would not be a great deviation from the mean as a result.

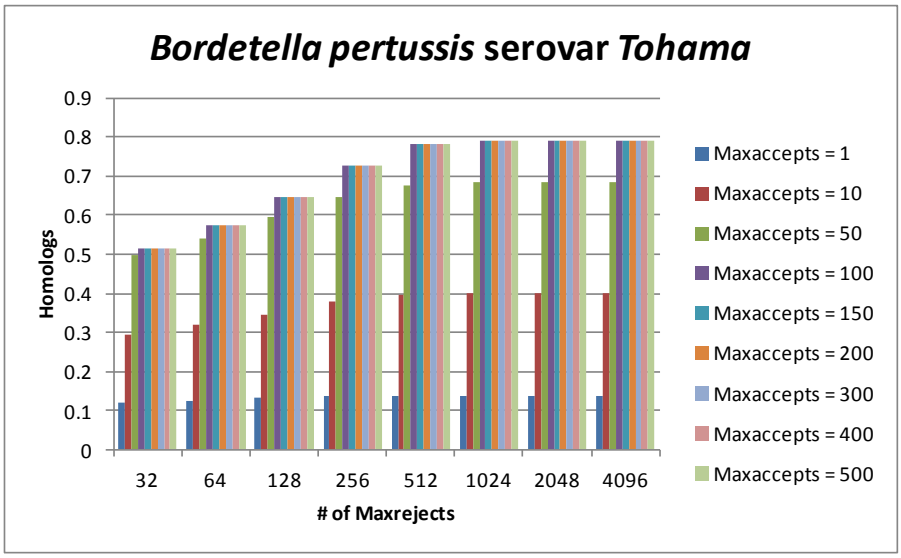


Figure A3.2.5: Number of putative homologs detected by USEARCH when different values of the –maxaccepts and –maxrejects options were specified; the genome of *E. coli* K12 was used as the query and the genome of *Bordetella pertussis* was used as the target. The number of matches increases as the number of target sequences tested increases but only up to a threshold of 100 –maxaccepts and 1024 –maxrejects. This is the same threshold that is seen for this organism when time was monitored. Results were normalized against the number of homologs detected by BLASTP. Error bars are not depicted as the numerical results show the same pattern and there would not be a great deviation from the mean as a result.

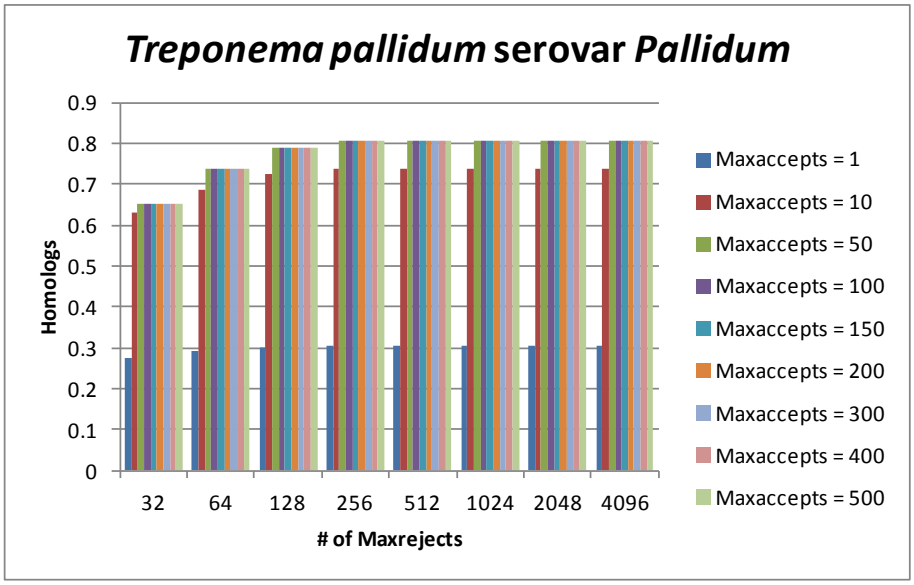


Figure A3.2.6: Number of putative homologs detected by USEARCH when different values of the –maxaccepts and –maxrejects options were specified; the genome of *E. coli* K12 was used as the query and the genome of *Treponema pallidum* was used as the target. The number of matches increases as the number of target sequences tested increases but only up to a threshold of 50 –maxaccepts and 256 –maxrejects. This is the same threshold that is seen for this organism when

time was monitored. Results were normalized against the number of homologs detected by BLASTP. Error bars are not depicted as the numerical results show the same pattern and there would not be a great deviation from the mean as a result.

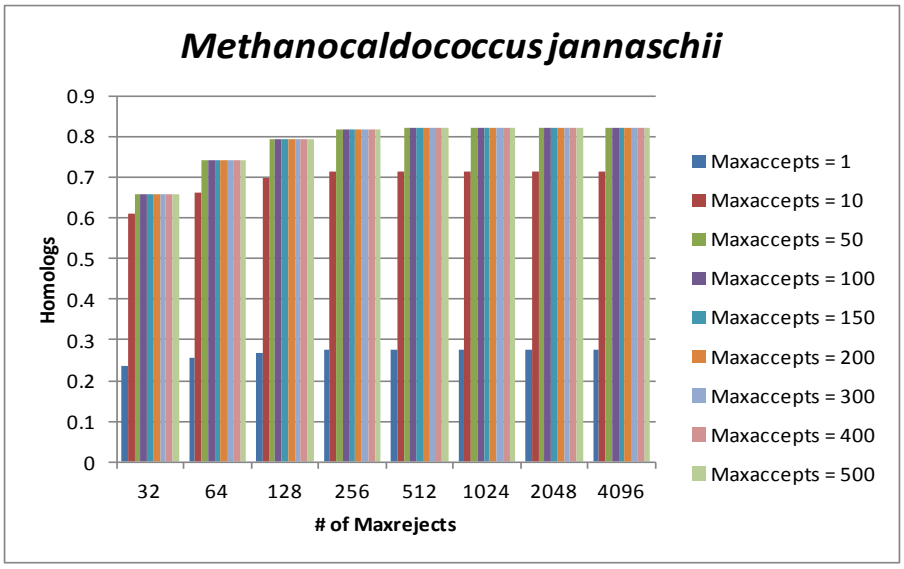


Figure A3.2.7: Number of putative homologs detected by USEARCH when different values of the –maxaccepts and –maxrejects options were specified; the genome of *E. coli* K12 was used as the query and the genome of *Methanocaldococcus jannaschii* was used as the target. The number of matches increases as the number of target sequences tested increases but only up to a threshold of 50 –maxaccepts and 256 –maxrejects. This is a slightly different threshold than what is seen for this organism when time was monitored. Results were normalized against the number of homologs detected by BLASTP. Error bars are not depicted as the numerical results show the same pattern and there would not be a great deviation from the mean as a result.

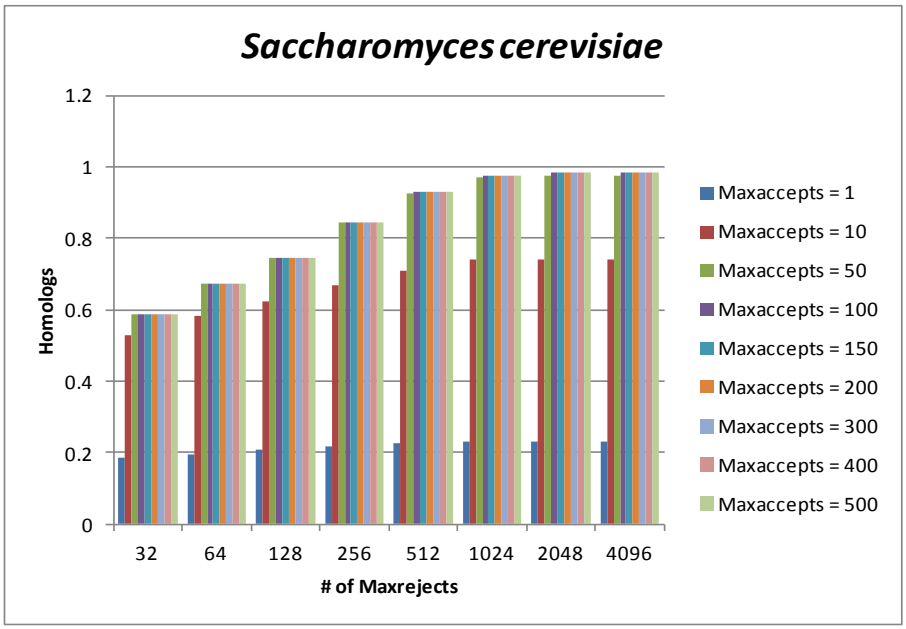


Figure A3.2.8: Number of putative homologs detected by USEARCH when different values of the –maxaccepts and –maxrejects options were specified; the genome of *E. coli* K12 was used as the query and the genome of *Saccharomyces cerevisiae* was used as the target. The number of matches

increases as the number of target sequences tested increases but only up to a threshold of 100 –maxaccepts and 1024 –maxrejects. This is a slightly different threshold than what is seen for this organism when time was monitored. Results were normalized against the number of homologs detected by BLASTP. In this case, at the threshold, a very similar number of homologs are detected by USEARCH as are detected by BLASTP. Error bars are not depicted as the numerical results show the same pattern and there would not be a great deviation from the mean as a result.

A3.3 Testing USEARCH Options: Ortholog Detection

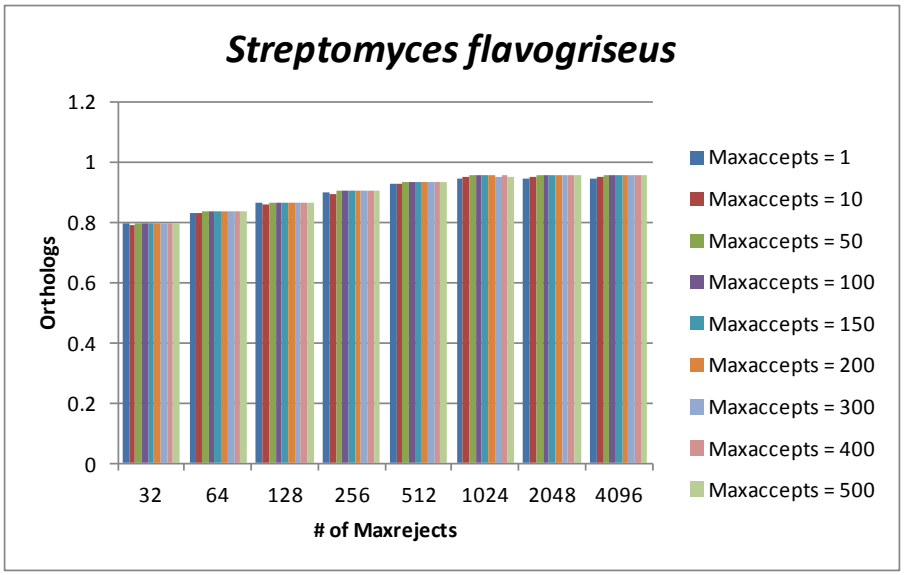


Figure A3.3.1: Number of putative orthologs detected by USEARCH when different values of the –maxaccepts and –maxrejects options were specified; the genome of *E. coli* K12 was used as the query and the genome of *Streptomyces flavogriseus* was used as the target. The number of orthologs found increases as the number of target sequences tested increases but only up to a threshold of 50 –maxaccepts and 1024 –maxrejects. This is a slightly different threshold than that seen for the homolog data. Results were normalized against the number of orthologs detected by BLASTP. There is a greater similarity in the number of orthologs detected by USEARCH (at threshold values) and BLASTP than homologs. Error bars are not depicted as the numerical results show the same pattern and there would not be a great deviation from the mean as a result.

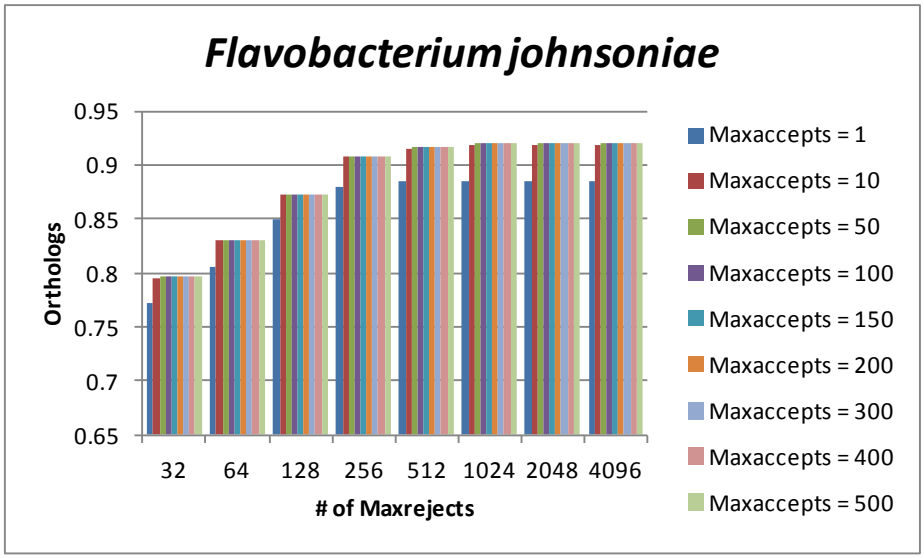


Figure A3.3.2: Number of putative orthologs detected by USEARCH when different values of the –maxaccepts and –maxrejects options were specified; the genome of *E. coli* K12 was used as the query and the genome of *Flavobacterium johnsoniae* was used as the target. The number of orthologs found increases as the number of target sequences tested increases but only up to a threshold of 50 –maxaccepts and 512 –maxrejects. This is a slightly different threshold than that seen for the homolog data. Results were normalized against the number of orthologs detected by BLASTP. There is a greater similarity in the number of orthologs detected by USEARCH (at threshold values) and BLASTP than homologs. Error bars are not depicted as the numerical results show the same pattern and there would not be a great deviation from the mean as a result.

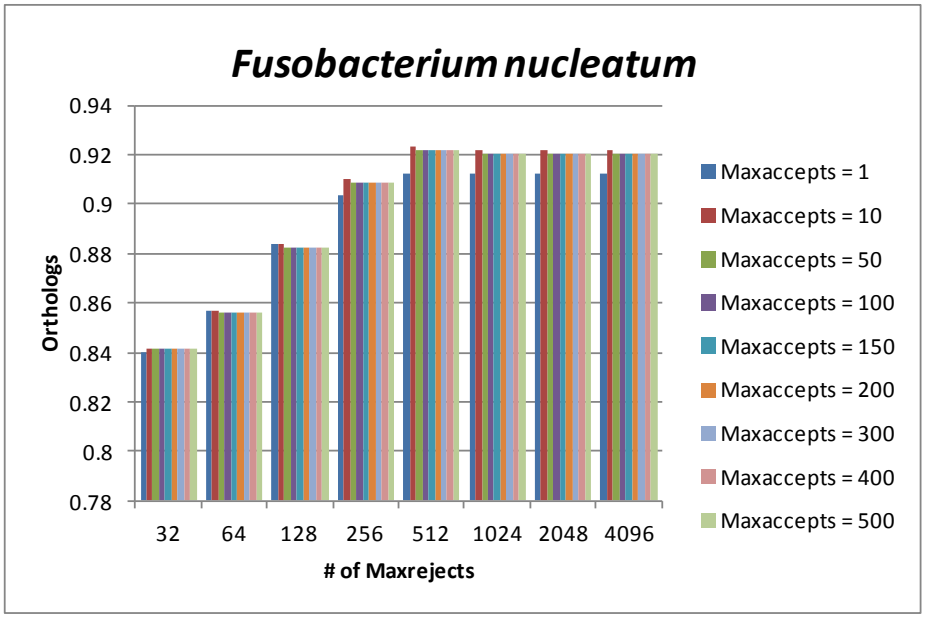


Figure A3.3.3: Number of putative orthologs detected by USEARCH when different values of the –maxaccepts and –maxrejects options were specified; the genome of *E. coli* K12 was used as the query and the genome of *Fusobacterium nucleatum* was used as the target. Here, the number of orthologs found is highest at 10 –maxaccepts and 512 –maxrejects; further, 10 –maxaccepts returns the highest number of orthologs at each corresponding –maxrejects value. However, a plateau is seen at 50 –maxaccepts and 1024 –maxrejects. This is a slightly different threshold than that seen

for the homolog data. Results were normalized against the number of orthologs detected by BLASTP. There is a greater similarity in the number of orthologs detected by USEARCH (at threshold values) and BLASTP than homologs. Error bars are not depicted as the numerical results show the same pattern and there would not be a great deviation from the mean as a result.

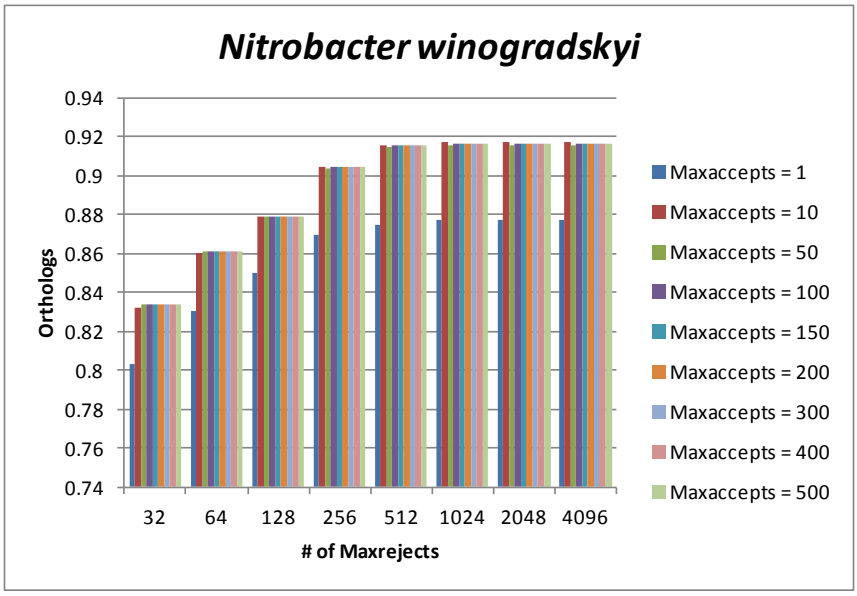


Figure A3.3.4: Number of putative orthologs detected by USEARCH when different values of the –maxaccepts and –maxrejects options were specified; the genome of *E. coli* K12 was used as the query and the genome of *Nitrobacter winogradskyi* was used as the target. When 256 –maxrejects and greater are specified, a –maxaccepts value of 10 returns the highest number of orthologs. However, a plateau is seen at 100 –maxaccepts and 1024 –maxrejects. This is a slightly different threshold than that seen for the homolog data. Results were normalized against the number of orthologs detected by BLASTP. There is a greater similarity in the number of orthologs detected by USEARCH (at threshold values) and BLASTP than homologs. Error bars are not depicted as the numerical results show the same pattern and there would not be a great deviation from the mean as a result.

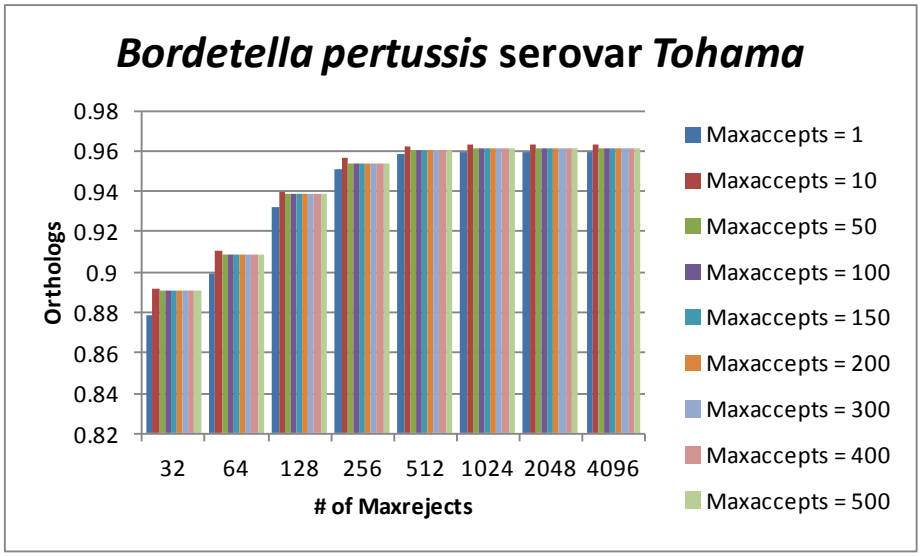


Figure A3.3.5: Number of putative orthologs detected by USEARCH when different values of the –maxaccepts and –maxrejects options were specified; the genome of *E. coli* K12 was used as the query and the genome of *Bordetella pertussis* was used as the target. Again, the largest number of orthologs is detected at a –maxaccepts value of 10 at every corresponding –maxrejects value. However, a plateau is seen at 50 –maxaccepts and 1024 –maxrejects. This is a slightly different threshold than that seen for the homolog data. Results were normalized against the number of orthologs detected by BLASTP. There is a greater similarity in the number of orthologs detected by USEARCH (at threshold values) and BLASTP than homologs. Error bars are not depicted as the numerical results show the same pattern and there would not be a great deviation from the mean as a result.

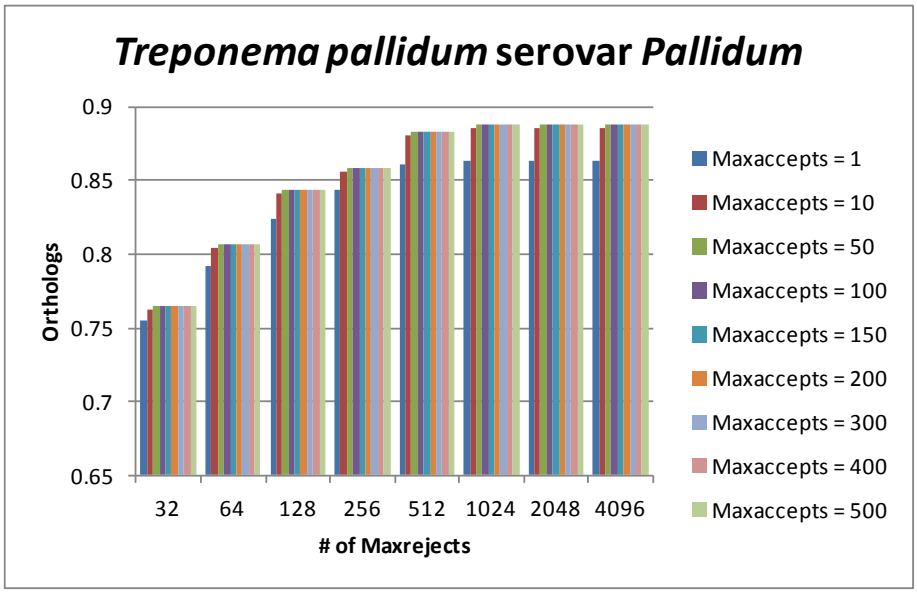


Figure A3.3.6: Number of putative orthologs detected by USEARCH when different values of the –maxaccepts and –maxrejects options were specified; the genome of *E. coli* K12 was used as the query and the genome of *Treponema pallidum* was used as the target. The number of orthologs found increases as the number of target sequences tested increases but only up to a threshold of 50 –maxaccepts and 1024 –maxrejects. This is a slightly different threshold than that seen for the homolog data. Results were normalized against the number of orthologs detected by BLASTP.

There is a greater similarity in the number of orthologs detected by USEARCH (at threshold values) and BLASTP than homologs. Error bars are not depicted as the numerical results show the same pattern and there would not be a great deviation from the mean as a result.

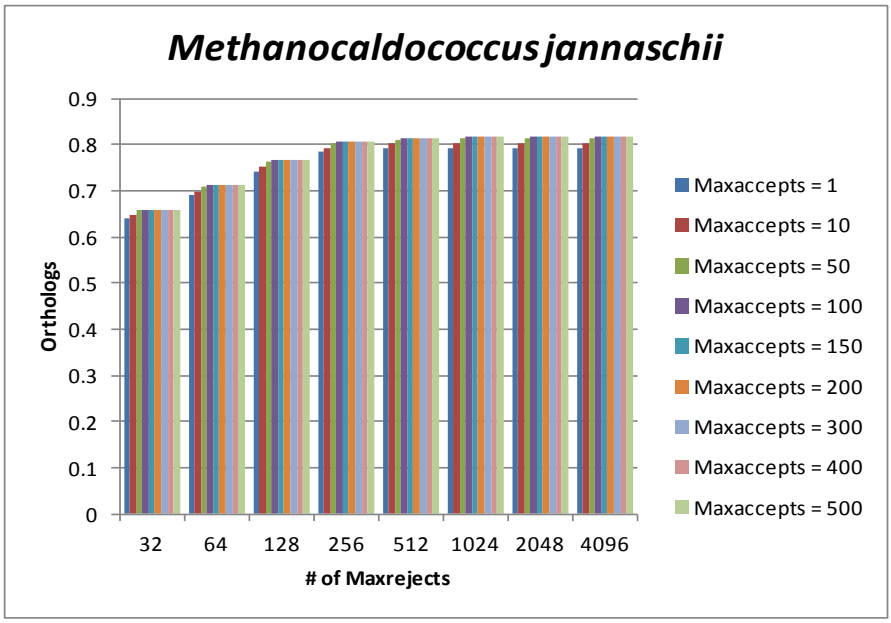


Figure A3.3.7: Number of putative orthologs detected by USEARCH when different values of the –maxaccepts and –maxrejects options were specified; the genome of *E. coli* K12 was used as the query and the genome of *Methanocaldococcus jannaschii* was used as the target. The number of orthologs found increases as the number of target sequences tested increases but only up to a threshold of 100 –maxaccepts and 1024 –maxrejects. This is a slightly different threshold than that seen for the homolog data. Results were normalized against the number of orthologs detected by BLASTP. There is a greater similarity in the number of orthologs detected by USEARCH (at threshold values) and BLASTP than homologs. Error bars are not depicted as the numerical results show the same pattern and there would not be a great deviation from the mean as a result.

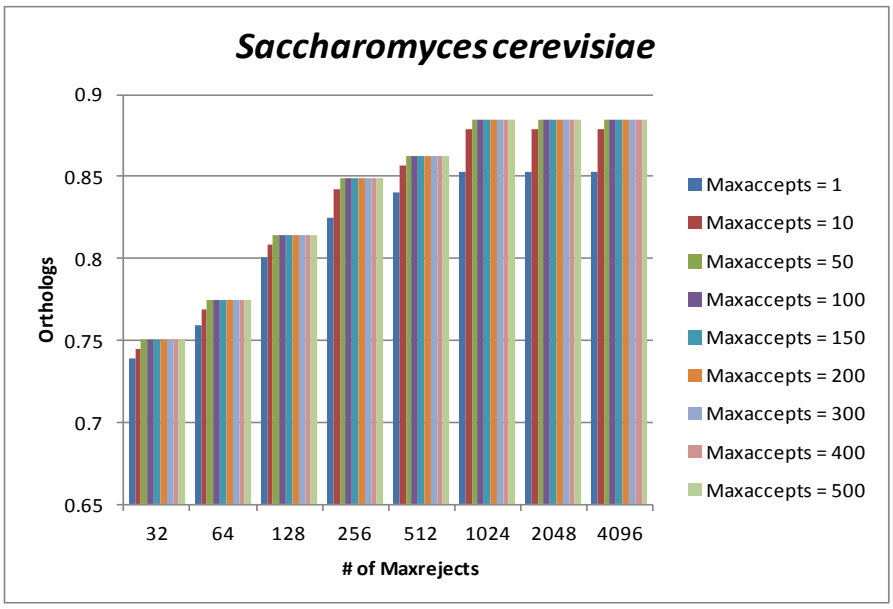


Figure A3.3.8: Number of putative orthologs detected by USEARCH when different values of the –maxaccepts and –maxrejects options were specified; the genome of *E. coli* K12 was used as the query and the genome of *Saccharomyces cerevisiae* was used as the target. The number of orthologs found increases as the number of target sequences tested increases but only up to a threshold of 50 –maxaccepts and 1024 –maxrejects. This is a slightly different threshold than that seen for the homolog data. Results were normalized against the number of orthologs detected by BLASTP. There is a greater similarity in the number of orthologs detected by USEARCH (at threshold values) and BLASTP than homologs. Error bars are not depicted as the numerical results show the same pattern and there would not be a great deviation from the mean as a result.

A3.4 Testing USEARCH Options: Ortholog Error Rates

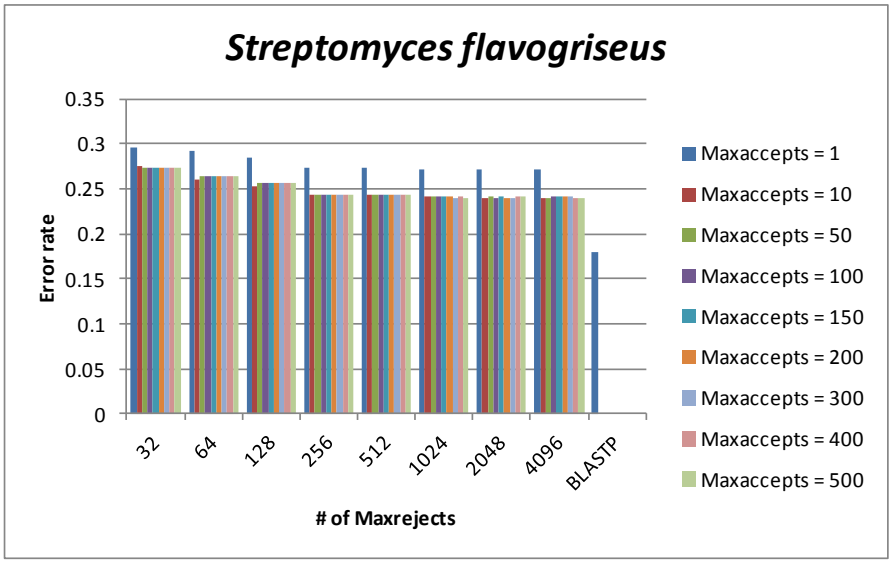


Figure A3.4.1: Error rate estimates associated with ortholog detection at each specified USEARCH option combination in comparison to BLASTP. The genome of *E. coli* K12 was used as the query and the genome of *Streptomyces flavogriseus* was used as the target. There are many mistakes

made at a –maxaccepts value of 1 regardless of the –maxrejects value specified, indicating that allowing only one hit does not provide an accurate list of correct orthologs. Error rates tend to decrease as a greater number of target sequences are tested; this makes sense as searching more sequences will increase the likelihood of finding a more accurate hit. As well, the same plateau effect is seen here as was observed with all other parameters tested. BLASTP is able to detect more orthologs of higher quality in this case than USEARCH. Error bars are not depicted as the numerical results show the same pattern and there would not be a great deviation from the mean as a result.

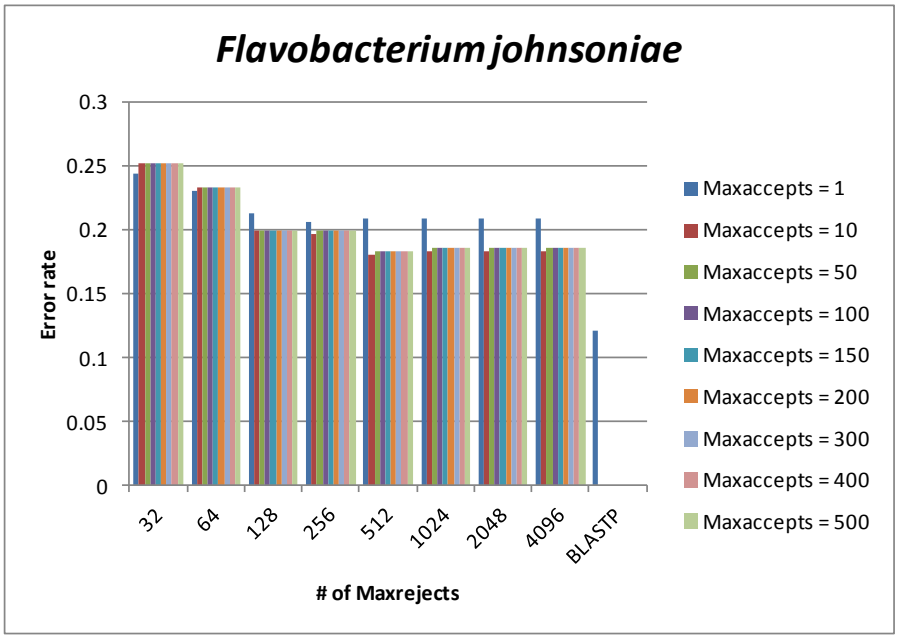


Figure A3.4.2: Error rate estimates associated with ortholog detection at each specified USEARCH option combination in comparison to BLASTP. The genome of *E. coli K12* was used as the query and the genome of *Flavobacterium johnsoniae* was used as the target. There is a high error rate associated with a –maxaccepts value of 1 regardless of the –maxaccepts value specified. This indicates that allowing only one hit does not provide an accurate list of correct orthologs. Error rates tend to decrease as a greater number of target sequences are tested; this makes sense as searching more sequences will increase the likelihood of finding a more accurate hit. As well, the same plateau effect is seen here as was observed with all other parameters tested. BLASTP is able to detect more orthologs of higher quality in this case than USEARCH. Error bars are not depicted as the numerical results show the same pattern and there would not be a great deviation from the mean as a result.

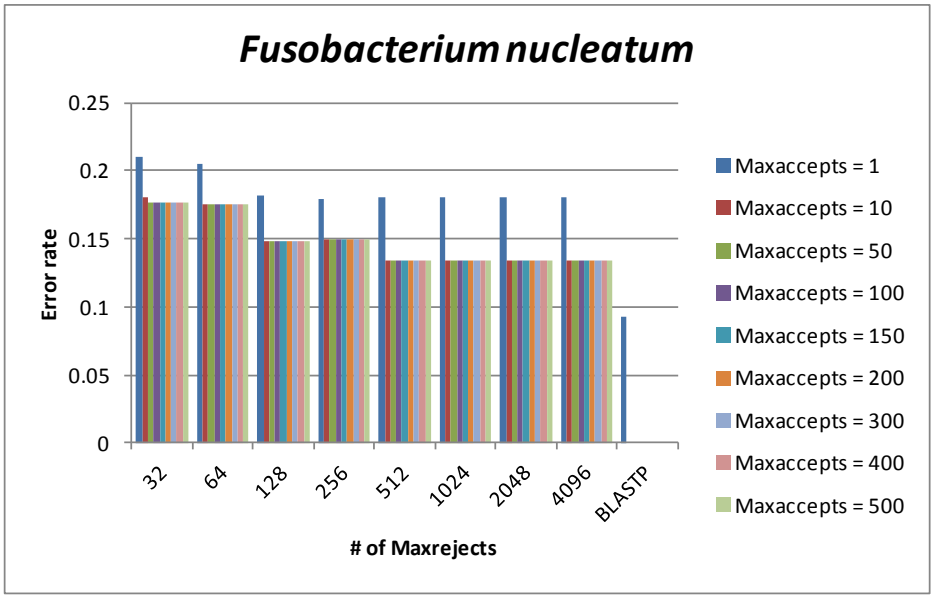


Figure A3.4.3: Error rate estimates associated with ortholog detection at each specified USEARCH option combination in comparison to BLASTP. The genome of *E. coli* K12 was used as the query and the genome of *Fusobacterium nucleatum* was used as the target. There is a high error rate associated with a –maxaccepts value of 1 regardless of the –maxaccepts value specified. This indicates that allowing only one hit does not provide an accurate list of correct orthologs. Error rates tend to decrease as a greater number of target sequences are tested; this makes sense as searching more sequences will increase the likelihood of finding a more accurate hit. As well, the same plateau effect is seen here as was observed with all other parameters tested. BLASTP is able to detect more orthologs of higher quality in this case than USEARCH. Error bars are not depicted as the numerical results show the same pattern and there would not be a great deviation from the mean as a result.

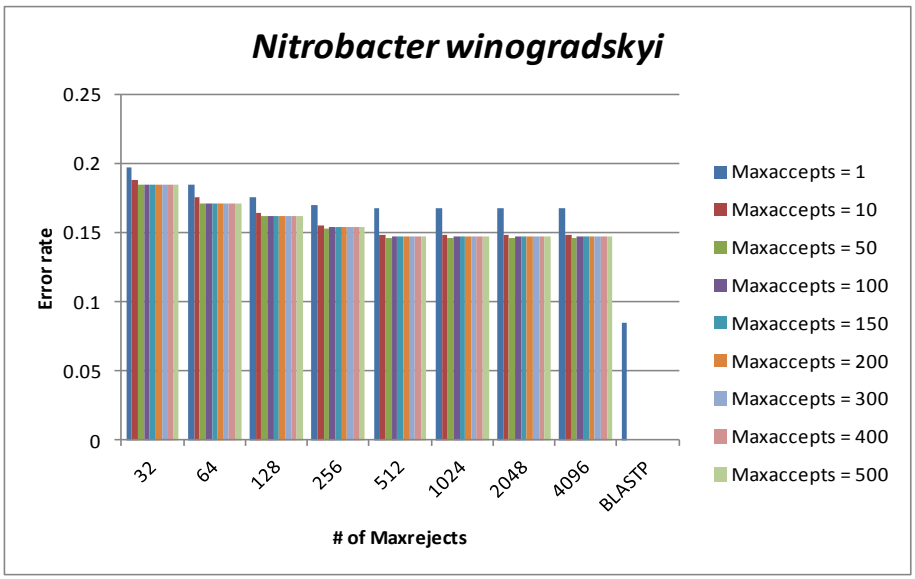


Figure A3.4.4: Error rate estimates associated with ortholog detection at each specified USEARCH option combination in comparison to BLASTP. The genome of *E. coli* K12 was used as the query and the genome of *Nitrobacter winogradskyi* was used as the target. There is a high error rate associated with a –maxaccepts value of 1 regardless of the –maxaccepts value specified. This

indicates that allowing only one hit does not provide an accurate list of correct orthologs. Error rates tend to decrease as a greater number of target sequences are tested; this makes sense as searching more sequences will increase the likelihood of finding a more accurate hit. As well, the same plateau effect is seen here as was observed with all other parameters tested. BLASTP is able to detect more orthologs of higher quality in this case than USEARCH. Error bars are not depicted as the numerical results show the same pattern and there would not be a great deviation from the mean as a result.

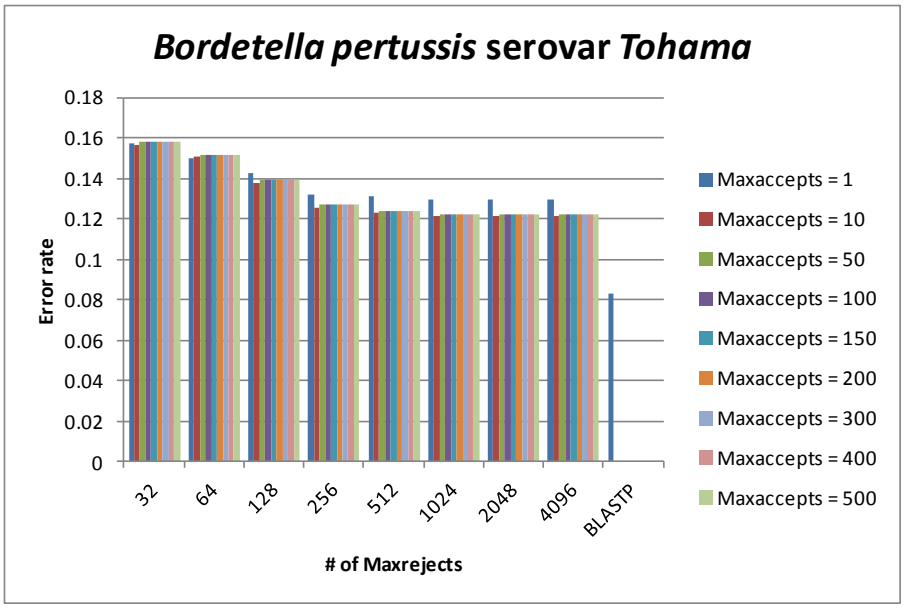


Figure A3.4.5: Error rate estimates associated with ortholog detection at each specified USEARCH option combination in comparison to BLASTP. The genome of *E. coli* K12 was used as the query and the genome of *Bordetella pertussis* was used as the target. There is a high error rate associated with a –maxaccepts value of 1 regardless of the –maxaccepts value specified. This indicates that allowing only one hit does not provide an accurate list of correct orthologs. Error rates tend to decrease as a greater number of target sequences are tested; this makes sense as searching more sequences will increase the likelihood of finding a more accurate hit. As well, the same plateau effect is seen here as was observed with all other parameters tested. BLASTP is able to detect more orthologs of higher quality in this case than USEARCH. Error bars are not depicted as the numerical results show the same pattern and there would not be a great deviation from the mean as a result.

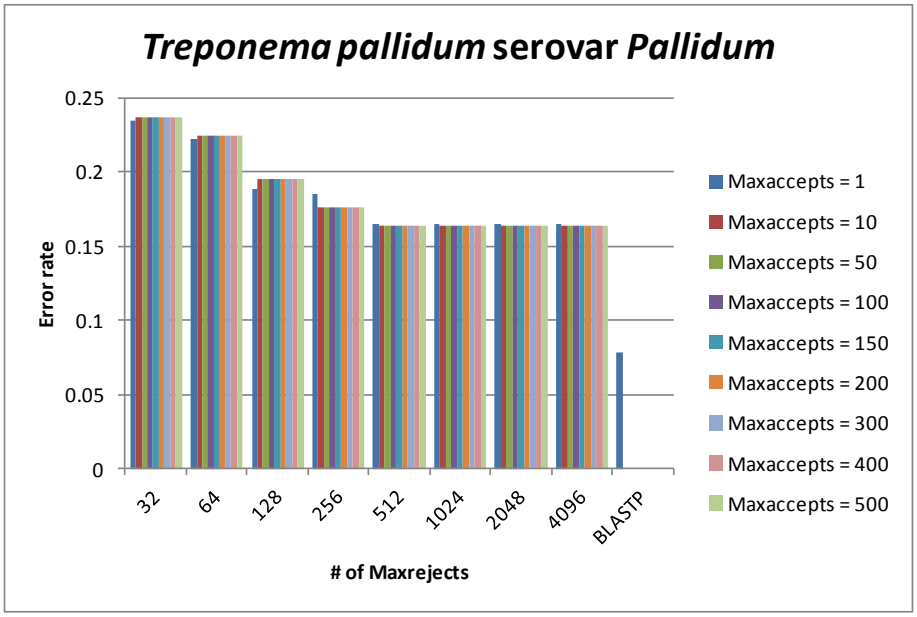


Figure A3.4.6: Error rate estimates associated with ortholog detection at each specified USEARCH option combination in comparison to BLASTP. The genome of *E. coli* K12 was used as the query and the genome of *Treponema pallidum* was used as the target. There is a high error rate associated with a –maxaccepts value of 1 regardless of the –maxaccepts value specified. This indicates that allowing only one hit does not provide an accurate list of correct orthologs. Error rates tend to decrease as a greater number of target sequences are tested; this makes sense as searching more sequences will increase the likelihood of finding a more accurate hit. As well, the same plateau effect is seen here as was observed with all other parameters tested. BLASTP is able to detect more orthologs of higher quality in this case than USEARCH. Error bars are not depicted as the numerical results show the same pattern and there would not be a great deviation from the mean as a result.

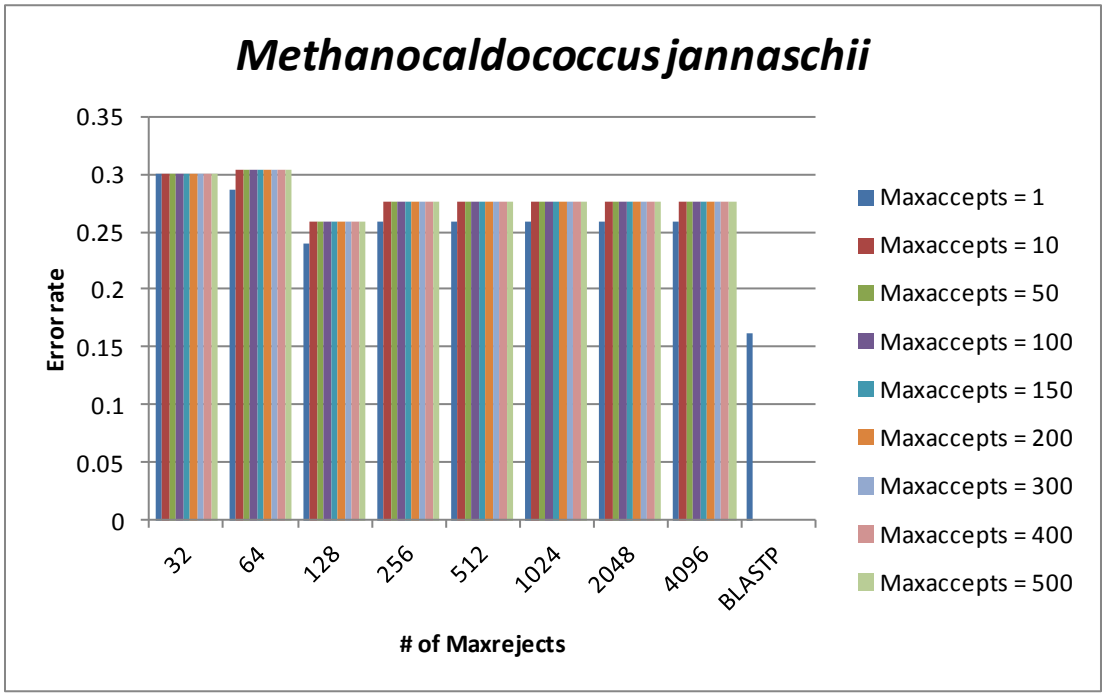


Figure A3.4.7: Error rate estimates associated with ortholog detection at each specified USEARCH option combination in comparison to BLASTP. The genome of *E. coli* K12 was used as the query and the genome of *Methanocaldococcus jannaschii* was used as the target. Error rates tend to decrease as a greater number of target sequences are tested; this makes sense as searching more sequences will increase the likelihood of finding a more accurate hit. As well, the same plateau effect is seen here as was observed with all other parameters tested. BLASTP is able to detect more orthologs of higher quality in this case than USEARCH. Error bars are not depicted as the numerical results show the same pattern and there would not be a great deviation from the mean as a result.