Targeted Optimization of Computational and Classification Performance of a Protein-Protein Interaction Predictor

by

Catalin Patulea

A thesis submitted to the Faculty of Graduate and Postdoctoral Affairs in partial fulfillment of the requirements for the degree of

Master of Applied Science

in

Electrical and Computer Engineering

Carleton University
Ottawa, Ontario

September 2011

Your file Votre référence
ISBN: 978-0-494-83046-8
Our file Notre référence
ISBN: 978-0-494-83046-8

The undersigned recommend to the
Faculty of Graduate and Postdoctoral Affairs
acceptance of the thesis

Targeted Optimization of Computational and Classification Performance of a Protein-Protein Interaction Predictor

Submitted by Catalin Patulea
in partial fulfillment of the requirements for
the degree of Master of Applied Science in Electrical and Computer Engineering

_____

Thesis Supervisor
Dr. James R Green

_____

Chair, Department of Systems and Computer Engineering
Dr. Howard Schwartz

2011, Carleton University

Abstract

Protein-protein interactions are an important aspect of cell structure because they are one of the primary mechanisms by which proteins carry out their functions in the cell. Organism-wide protein-protein interaction screens can provide a tremendous amount of information on protein function and cell structure, but experimental techniques become costly and error-prone at this scale. Computational prediction of protein-protein interactions can overcome some of these limitations, provided that the predictor is computationally efficient and sufficiently accurate.

In this thesis, we contribute two improvements to the leading sequence-based interaction predictor: First, using improved data structures and a space-time trade-off, we optimized several often-executed fragments of the algorithm. This results in speedups ranging from 8.1 to 14.5 on sets of protein pairs from *S. cerevisiae* and *H. sapiens*, allowing exhaustive probing of all interactions in *H. sapiens* in 20 days on a 28-core cluster. Second, we modified the predictor to take into account annotations of sequence regions which re-occur in the proteome but are unlikely to be involved in protein-protein interactions, such as signal peptides and transmembrane regions. This increased sensitivity from 2.17% to 2.25% in *S. cerevisiae* and from 1.26% to 4.23% in *H. sapiens*. However, an extension of this idea, which uses only the protein sequences to measure the re-occurrence of sequence windows and adjust their score accordingly, achieved larger gains in classification performance, improving classifier sensitivity from 2.17% to 3.05% in *S. cerevisiae* and from 1.26% to 16.19% in *H. sapiens*.

Acknowledgements

I would like to thank my supervisor, Prof. James Green, for supporting me and guiding this research. Many thanks also to Sylvain Pitre, who was always available for brainstorming and who provided much-needed perspective at all stages of the work. Thank you to all the other PIPE authors, whose initial work represents the seed for this research.

Thank you to NSERC, OGS, and the Department of Systems and Computer Engineering, who provided funding.

Finally, thank you to my parents, who encouraged me to continue my studies and who supported me during this time.

Table of Contents

List of Tables

List of Figures

1 Introduction

1.1 Background

Proteins are biopolymers of amino acids which have been implicated in virtually every process in the cell, and whose amino acid sequences are encoded by the corresponding genes of a given organism: In DNA transcription and translation, proteins selectively activate certain genes depending on the cell state and also convert the chosen genes into new proteins from raw material (individual amino acids). In energy conversion, they form the machinery which converts proton gradients between the interior and exterior of the cell into chemical energy which can be used by other cell components. In signal transduction, proteins are the membrane receptors which activate interior messengers (also proteins) when the exterior of the cell is stimulated with specific ligands. Finally, in waste processing, proteins decompose other proteins when they are no longer needed by the cell, recycling amino acids for the construction of new proteins during DNA translation.

Due to their central and diverse role in the cell, proteins have also been found at the core of many diseases. For example, Alzheimer's and Huntington's are neurodegenerative diseases in which misfolded proteins accumulate in the brain [1], reducing the supply of the native, functional form of the protein. The HIV virus (which itself is contained inside a protein coat) targets host cells and gains access to them through proteins embedded in the host cell's membrane [2]. On the other hand, some antiviral drugs are specific inhibitors of viral proteins known to be required for viral activity [3]. Therefore, the study of protein function is essential for understanding the

functioning of the cell in general, and is especially relevant for rational, targeted treatment for a vast number of conditions.

The sequence of amino acids of a protein is encoded by a gene from that organism's genome, so it is intuitive to suspect that the number of genes is the main determining factor in an organism's biological complexity, and that this complete set of protein products should be the primary target when studying an organism as a whole. However, when we consider that humans have between 20,000 and 25,000 genes [4], the worm *Caenorhabditis elegans* has approximately 19,000 genes [5], and rice has between 32,000 and 50,000 genes [6], it becomes clear that this number cannot be the only factor. Recently, it has been suggested that the number of *interactions* between proteins better accounts for organism complexity (though other factors such as alternative splicing and non-coding RNA also contribute), and this is reflected by estimated numbers of interactions which are more than 3 times greater in *H. sapiens* than in *C. elegans* [7].

This suggests that the study of protein-protein *interactions* is critical when seeking a comprehensive understanding of the functioning of a biological organism. Experimental "wet lab" techniques for testing protein-protein interactions are capable of testing two individual proteins for interaction. However, this is a time-consuming and costly process, especially when attempted at a large scale. Some methods are fraught with erroneous results due to intrinsic limitations of the experimental method, and may be particularly weak on specific types of proteins (e.g. membrane proteins). High-throughput methods produce results for batches of protein-protein interaction experiments faster than sequences of individual experiments, and have resulted in data sets of observed interactions for entire organisms [8]. However, the overlap between

independent experiments has historically been low, calling the reliability of the results into question [9].

Computational protein-protein interaction assays promise more efficient and flexible probing of protein-protein interactions, but have their own associated caveats. Computational methods can be largely grouped into two categories, according to the level of protein structure used to represent the input proteins, which also determines the type of protein data required. In protein docking, the three-dimensional structures of the two query proteins are tested for surface complementarity and electrostatic interactions using molecular dynamics techniques. This requires that the structures for both proteins are known, however this data is difficult to obtain. As of May 2011, there are only 1,102 structures of *H. sapiens* proteins in the largest protein structure database PDB [10] (compare to the estimated 20,000 genes in *H. sapiens*). Additionally, molecular dynamics simulations are very computationally-intensive, requiring many months of processing on large distributed clusters. Protein docking is not a practical tool for probing protein-protein interactions on a large scale.

On the other hand, protein-protein interactions can be predicted from protein sequence alone [11]. With the advent of whole-genome sequencing, datasets of organism proteomes are easier to obtain, independently confirmed by more researchers, and therefore increasingly reliable. The human genome has been completely sequenced since 2001 [12] [13], and as of 2009 the genomes of more than 1000 other organisms have been sequenced [14]. Sequence-based methods also usually require experimental data on protein-protein interactions as training data, but interaction data are more abundant than structure data. Data sets on the order of 35,000 interactions for *H. sapiens* are not

uncommon and, while representing a very small fraction of the total estimated interactions, are sufficient to train a machine learning model for protein-protein interaction prediction. Given a well-trained model, sequence-based computational methods represent a cost-effective method of exploring protein-protein interaction networks, using data which is available today. They also open the door to exhaustive probing of interactions between all pairs of proteins within an organism's proteome. However, the set of all potential interactions (all pairs of proteins) is orders of magnitude larger than the sets of interactions used by most authors to test their methods. This suggests that the computational performance of their methods may be a bottleneck in their methods' applicability to all-to-all screens. Computational methods do however scale seamlessly with available computational resources, which are steadily increasing per unit of cost.

In machine learning terms, protein-protein interaction (PPI) predictors are binary classifiers. The model takes as input two protein pairs and outputs a binary decision for the interaction between the proteins. The input proteins are represented by their amino acid sequences, but the classifier may also require additional auxiliary data, such as sequence annotations. In this context, the performance of the classifier on a given test set is measured in terms of specificity and sensitivity, which are related to the rates of false positive and false negative errors the model makes. For a protein-protein interaction predictor to be applicable to whole-organism interaction screens ("all-to-all" screens), it is crucial for the model to make very few false positive errors (specificity above 99%) because the ratio of true interactions to all possible pairs of proteins is estimated to be very low. Based on estimates of true interactions [7], the ratio is approximately 1:370 in

H. sapiens. For example, a classifier with an apparently good performance of 90% specificity and 90% sensitivity, applied to a sample of human protein pairs, would result in a precision of only 2.4%. (Details on how this is calculated are given in section 2.1.) This means that, on average, only 2.4% of protein pairs predicted to interact by the model do in fact interact, while the rest are false positives.

Recently, an independent survey of sequence-based protein-protein interaction prediction methods [15] tested 4 interaction methods, including a method developed by the bioinformatics group at Carleton University. The author tested the methods on two test sets, with ratios of positive to negative interactions of 1:10 and 1:100, and found that the classifiers achieved a higher precision on the 1:10 set with no change to the model, highlighting the need for test sets with more realistic ratios (closer to the ratios expected in the cell). The method developed at Carleton University, the Protein-protein Interaction Prediction Engine (PIPE) [16] [17], consistently outperformed the other 3 methods for interactions in both *S. cerevisiae* and *H. sapiens* and on both test sets. The author comments that applying the methods to test sets with even higher ratios became computationally prohibitive. Additionally, the author's consensus method was able to slightly outperform PIPE.

1.2 Statement of the Problem

Park's review of protein-protein interaction prediction methods [15] provides evidence that PIPE is currently the leading single sequence-based PPI predictor in its field. However, the method still has weaknesses which should be addressed.

First, the consensus combination of 4 methods slightly outperformed PIPE in both *S. cerevisiae* and *H. sapiens*. This suggests that while the PIPE model is strong for the

majority of protein-protein interactions, it still makes erroneous predictions for some fraction of protein pairs. Thus, the sources of error should be investigated and, whenever possible, the model should be adjusted to account for them. Because of the importance of operating at high specificity, priority should be given to reducing false positive errors.

Second, Park comments that testing on data sets with larger ratios of negatives to positives, which better represent the ratio expected in the cell, would be computationally prohibitive. This suggests that the problem of computational performance of protein-protein prediction methods has not yet been addressed. At the same time, model design and optimization requires many iterations of model training and testing, and large all-to-all interaction screens must process numbers of pairs which are orders of magnitude larger than the size of a typical test set. Therefore, improvements in computational performance are crucial for thorough model design, fair model testing, and for applicability to large-scale interaction screens.

1.3 Contributions

1.3.1 Computational Performance

We analyzed the computational performance of the current implementation of PIPE, with consideration to organism-specific proteome parameters which may affect algorithm performance. Because PIPE primarily consists of memory accesses to its input data, rather than arithmetic operations, we suspected that its performance is limited by how efficiently it uses the memory subsystem of the processor. We hypothesized that data structures which reduce the cost and number of memory accesses to the input data can speed up the algorithm. To verify this hypothesis, we constructed an algorithm which uses a more CPU cache-efficient data structure for some of the inputs, and which applies

a time-space trade-off to reduce the number of memory accesses required to compute the same result, in a particular fragment of the algorithm. These modifications achieved a 8.9-fold speedup on total runtime on a *S. cerevisiae* test set, and a 14.5-fold speedup on total runtimeon a *H. sapiens* test set (up to 80-fold for individual pairs).

1.3.2 Classification Performance

We explored potential sources of errors in the PIPE model from protein regions which re-occur in the proteome but are unlikely to participate in PPIs. We first considered two specific types of re-occurring protein regions for which sequence annotations are widely available: signal peptides and transmembrane regions. Then, we extended this idea to the re-occurrence of arbitrary sequence windows from the query proteins.

First, we hypothesized that signal peptides and transmembrane regions do not overlap with PPI-mediating protein regions. We considered both experimental and predicted protein regions and compared them to reference binding sites from two experimental databases. Specifically, we hypothesized that for any given amino acid in a protein, the events "amino acid is in a signal peptide/transmembrane region" and "amino acid is in a binding site" are negatively correlated, for any combination of experimental/predicted signal peptide/transmembrane region and using binding sites from either database. We tested each of these hypotheses using a left-tailed Fisher's exact test. The null hypotheses of independence between the two events were rejected in each case with a P-value of at most 8.2×10^{-3}. Therefore, we found that signal peptides and transmembrane regions are negatively correlated with binding sites.

We suspected that signal peptides and transmembrane regions contribute to noise in the PIPE matrix, because they re-occur but do not mediate PPIs, therefore increasing the PPI prediction false positive rate. We hypothesized that masking these regions prior to classification would reduce the false positive rate of PIPE (or equivalently, increase the sensitivity at the same specificity, see section 2.1). We constructed models which perform this masking using several different sequence annotation sets (experimental/predicted, signal peptides/transmembrane regions) and measured their classification performance on *S. cerevisiae* and *H. sapiens* test sets. At 99.95% specificity, the sensitivity of the classifier in *S. cerevisiae* increased from 2.17% to at most 2.25% (experimental transmembrane annotations). The sensitivity of the classifier in *H. sapiens* increased from 1.26% to at most 4.23% (experimental signal peptides).

We generalized this idea to arbitrary sequence windows from the query proteins, suspecting that *any* frequently re-occurring sequence window may contribute noise to the corresponding area of the PIPE output matrix, thereby increasing the false positive rate. We hypothesized that by normalizing the output score according to the uniqueness of the query sequence windows we can reduce this noise and improve classification performance of PIPE. We constructed a modified model which takes sequence window uniqueness into account, and tested this model on the same *S. cerevisiae* and *H. sapiens* test sets as in the previous tests. At 99.95% specificity, the sensitivity of the classifier in *S. cerevisiae* increased from 2.17% to 3.05% and in *H. sapiens* from 1.26% to 16.19%.

1.4 Organization of Thesis

Chapter 2 briefly introduces pattern recognition as it applies to protein-protein interaction prediction and describes several existing predictors, including a detailed

description of the PIPE method. Chapter 3 analyzes the computational performance of the current PIPE algorithm, proposes modifications and reports performance measurements for the new algorithm. Chapter 4 similarly describes three sources of error in the PIPE model, proposes modifications to the model and measures the classification performance of the new model. Finally, Chapter 5 summarizes the work presented in this thesis, and provides avenues for investigation for further improvement of PIPE.

2 Previous Work

2.1 Introduction to Pattern Classification

A protein-protein interaction predictor is a classifier which, given a pair of proteins, outputs a score indicating the likelihood of an interaction between those two proteins. A threshold can be applied to this score to obtain a single binary decision for a given protein pair. For a given threshold, a predictor classifies each pair of proteins as either interacting (positive prediction) or non-interacting (negative prediction).

Training and testing the model requires a set of protein pairs of known class, including both interacting and non-interacting pairs. Some classifiers, such as PIPE, use only positive pairs for training, but both types of pairs for testing. During training, the model learns which features are informative with respect to the class of each training instance. During testing, the classifier is applied to a test instance and the threshold is applied to result in a single binary output which represents the predicted class of the test instance. The instance's true class is then compared to the prediction made by the classifier. If we apply this procedure on the entire test set for a given score threshold, there are four possible cases (Figure 1):

- True positive: positive instance which was above the threshold (correctly predicted)
- False positive: negative instance which was above the threshold (incorrectly predicted as positive, or type I error)
- False negative: positive instance which fell below the threshold (incorrectly predicted as negative, or type II error)

- True negative: negative instance which fell below the threshold (correctly predicted)

By varying the score threshold, we can control a trade-off between false positive errors and false negative errors.

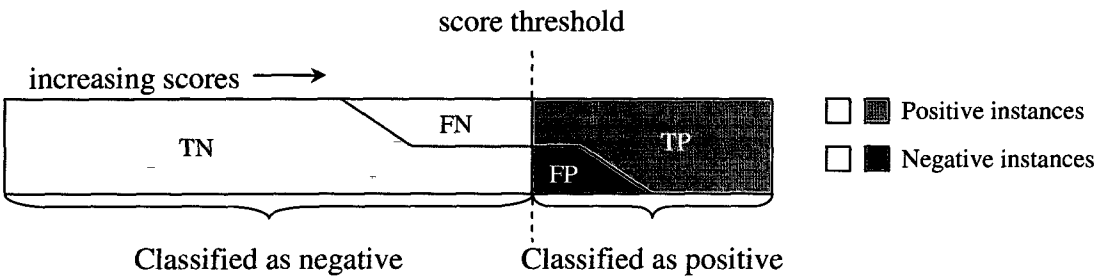


Figure 1: The four categories of test instances by classifier output score

The test set contains a total of P positive and N negative. The total counts of true positives (TP), false positives (FP), false negatives (FN) and true negatives (TN) for a given score threshold allow us to quantify the number and type of errors the classifier makes. In particular, errors on the positive set are measured using the true positive rate (TPR = TP / P), while errors on the negative set are measured using the false positive rate (FPR = FP / N). These are estimates of the probability that the classifier will correctly classify an arbitrary unseen positive instance, or to incorrectly classifier an arbitrary unseen negative instance, respectively.

From true positive rate (TPR) and false positive rate (FPR) we can derive several other measures of classifier performance. Sensitivity (also known as recall) (1) and specificity (2) are directly related to TPR and FPR and measure the performance on the positive and negative set, respectively. For protein-protein interactions, high specificity, or low false positive rate, is very important due to the high ratio of negatives to positives since most protein pairs would not be expected to physically interact.

$$S_n = TPR = \frac{TP}{P} = \frac{TP}{TP+FN} = Recall \quad (1)$$

$$S_p = 1 - FPR = 1 - \frac{FP}{N} = \frac{TN}{TN+FP} \quad (2)$$

To quantify classifier performance on the test set as a whole, there are other measures such as accuracy (3) and precision (4).

$$Acc = \frac{TP+TN}{P+N} = S_n \cdot \frac{P}{P+N} + S_p \cdot \frac{N}{P+N} \quad (3)$$

$$Prec = \frac{TP}{TP+FP} = \frac{1}{1+\frac{FP}{TP}} = \frac{1}{1+\frac{1-S_p}{S_n}\frac{N}{P}} \quad (4)$$

Accuracy is simply the ratio of correct predictions to the total number of instances. Precision is the ratio of correctly predicted instances to the total number of positive predictions made by the classifier. Note that accuracy and precision can be rewritten in terms of sensitivity, specificity, and the ratios of positives to negatives in the test set. While they can be very useful intuitive tools for measuring classifier performance, they are biased by the class ratio chosen by the experimenter for their test set. This is emphasized when the true ratio is expected to favour one class (such as in PPI where the ratio of all protein pairs to positive pairs in the cell is on the order of 350 negatives to 1 positive), but the test set is balanced (1:1). In the PPI example, if the test set is balanced, then specificity (S_p) and sensitivity (S_n) have equal contribution to the expected accuracy (3) because they are weighted by $\frac{P}{P+N} = \frac{N}{P+N} = \frac{1}{2}$. However, because the true N:P ratio is much greater, the accuracy on a random sample of protein pairs from the set of all pairs will be dominated by the specificity (S_p) term, and this effect will be exacerbated by greater N:P ratios. A classifier with good sensitivity and mediocre specificity will appear to have good accuracy on the balanced test set, but will have much worse accuracy when it is used on a random sample of pairs. Thus, accuracy is biased by

the N:P ratio in the test set, leading to overestimation of expected performance and the inability to directly compare accuracy values between different studies. Ideally, one would use a test set with the same ratio as the ratio found in the cell, but the true value of this ratio is not known. In practice, classifier performance is often reported by researchers in terms of accuracy, precision and recall, but on test sets with non-standardized (sometimes unspecified) negative to positive ratios, leading to measured performance which cannot be directly compared with other studies. When possible (i.e. when the class ratio was specified by the experimenter), we attempt to derive equivalent sensitivity and specificity values for quantitatively comparing different methods of PPI prediction. Sensitivity and specificity values, when taken together, are unbiased by the N:P ratio of the test set, and thus provide a fair comparison between studies.

The score threshold provides a trade-off between errors on the positive set (false negatives) and errors on the negative set (false positives). As we increase the threshold, fewer negative instances are misclassified as positive, but there is also an accompanying increase in positive instances misclassified as negative, and vice-versa. Thus, by varying the score threshold, it is possible to achieve arbitrary values of either sensitivity or specificity. Better models are those which can achieve increased sensitivity *at the same given sensitivity*, or, equivalently, increased sensitivity *at the same given specificity*.

The receiver operating characteristics (ROC) curve [18] allows us to visualize the trade-off between sensitivity and specificity. While varying the score threshold, from the highest instance score (reject all test instances) to the lowest score (accept all instances), specificity is plotted on the X axis and sensitivity on the Y axis. Some examples of ROC curves are shown in Figure 2, with the full range of specificity (left) and a zoom of the

region of specificity $\geq 90\%$ (right). Curve (a) represents a good classifier because of the high initial slope of the curve. This means that a small compromise in specificity provides a large improvement in sensitivity. Curve (b) requires a larger reduction in specificity to achieve the same improvement in sensitivity. For curve (c), the loss in specificity is exactly equal to the improvement in sensitivity (slope of the curve is 1). This represents a classifier which outputs a random classification and is the worst ROC curve.

Precision-recall curves can also be used to visualize this trade-off, but as previously shown, precision is biased by the class imbalance in the test set. Ideally, one would calculate precision-recall curves tailored to the specific ratio of negatives to positives expected to be observed in the target application of the method. In our case, the ratio of negative to positive interactions in the cell is not known with certainty and is also likely to vary by organism, making sensitivity-specificity curves a less biased method of reporting classifier performance than precision-recall curves.

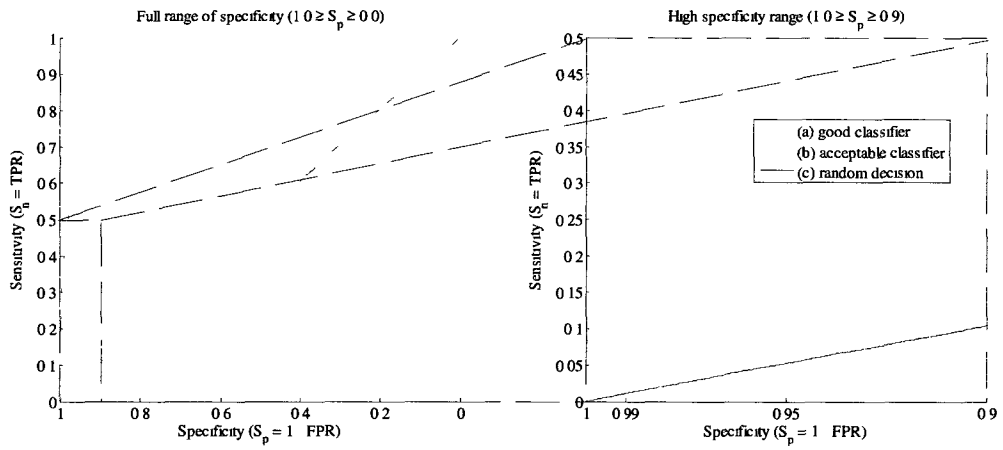


Figure 2: Sample receiver operating characteristics (ROC) curves. The left plot shows ROC curve across the full range of specificity and sensitivity ($1.0 \geq S_p \geq 0.0$). The right plot shows detail of the same ROC curve at high specificity ($1.0 \geq S_p \geq 0.9$ and $0.0 \leq S_n \leq 0.5$).

To be useful at a proteome scale, PPI classifiers must operate at very high specificity and must be evaluated in that region (far left side of the ROC curve graphs). In *H. sapiens*, there are approximately 22,000 known proteins, giving a total of approximately 242 million (unordered) pairs. Compare this to estimates placing the number of true interactions around 650,000 [7]. Even to achieve a precision of 50%, admitting half of positive predictions as errors (TP = FP = P), would require a false positive rate of $FPR = \frac{FP}{N} = \frac{650{,}000}{242{,}000{,}000 - 650{,}000} = 2.7 \times 10^{-3}$ (0.27 %), or a specificity of 99.73%. This simple calculation assumes all positives will be correctly predicted (Recall = 100%, or TP / P = TPR = S_n = 1.0); this is of course not usually the case. If the sensitivity is too low, the number of predicted true positives will be overwhelmed by the number of false positives (even though the false positive *rate* is very small), making the classifier outputs not very useful. Therefore, at the proteome-scale, the classifier must operate at very high specificity, and have reasonable sensitivity in that range.

2.2 Protein-Protein Interaction Prediction from Structure

Protein-protein interactions were initially studied using 3-D structures of proteins in complex. In 1996, Jones and Thornton [19] published an in-depth analysis of trends in structural features of 59 solved 3-D structures of several types of protein complexes. They found that certain features, in particular residue accessible surface area and residue propensities, were statistically biased in interface residues compared to non-interface surface residues. This was later used, along with other features, to predict protein interfaces from structure [20] [21] [22]. These methods are useful for guiding experimental assays on a specific pair of proteins. For example, the experimenter may focus their deletion or mutation experiments on the protein regions predicted from structure to be the protein interface. This increases the odds of a successful experiment compared to probing protein regions at random. However, for a single given protein, these methods do not identify all interaction partners from the pool of proteins present in the organism and thus are not useful for exploration of protein interaction networks.

By integrating structure information for two proteins simultaneously, existing methods have shown the ability to predict free energy on binding of certain known complexes (a measure of the affinity of the complex) [23], to predict novel protein complexes in *S. cerevisiae* [24], and to augment existing sequence-based predictions [25]. However, these methods depend on the availability of structure data for the proteins under consideration, while this data may be scarce. As of May 2011, there are only 1,102 structures for *H. sapiens* in the largest protein structure database, PDB [10], compared to the estimated 20,000 proteins in the organism. On the other hand, binary interaction data indicating only whether two given proteins were found to interact, which can be used to

identify complexes, is easier to obtain and therefore more abundant than protein structure data. In fact, it is common to find sets of 40,000 known interacting *H. sapiens* proteins.

2.3 Protein-Protein Interaction Prediction from Sequence

The protein structure availability problem can be overcome by developing models of protein interfaces from sequence data, which is much more widely available. This abundance has also given rise to a variety of derived data, such as orthologies between genes in different organisms, protein domains with known function [26] [27], and other sequence annotations [28]. The many possible combinations of sequence features, data pre-processing and machine learning models give rise to many types of sequence-based PPI prediction methods.

Initial attempts at *interface* prediction from sequence relied on simple observations such as overrepresentation of proline residues near interaction sites [29] and only determined which residues are likely to interact, without identifying the interaction partner protein. Subsequently, physicochemical properties of residues which can be inferred from sequence alone, such as hydrophobicity and charge, were used to predict linear stretches of sequence which are protein interfaces [30] [31] [32] [33]. While these methods cannot predict specific partners of the interfaces, they can be used to guide experimental assays into specific pairs of interacting proteins, and the types of sequence features introduced in this context have persisted in more complex methods of protein-protein interface prediction.

Proteins also exhibit conserved sequence regions which are sometimes referred to as protein domains. These are sequence patterns which occur in many proteins and across different organisms, and which have been determined to have a particular function. There

are well-established machine learning models (e.g. hidden Markov models) which are capable of detecting protein domains and other sequence features on newly-sequenced proteins, and databases which aggregate sequence annotations from many sources [27]. In practice, these sequence annotations have already been determined at the proteome scale for a number of well-studied organisms such as *S. cerevisiae* and *H. sapiens*.

Sprinzak and Margalit [34] termed these sequence patterns "sequence-signatures" and measured overrepresentation of sequence-signatures in protein pairs experimentally known to interact. This could be applied to interaction prediction between new protein pairs by considering the over or under-representation of pairs of sequence-signatures present in the new proteins. They found that some pairs of sequence signatures had a log-odds value as high as 12.16 compared to random occurrence, and the most frequent signature pair had a log-odds value of 2.52. In a leave-one-out validation of their method as applied to PPI prediction, they report sensitivity as high as 97%, but they defer estimation of specificity until an experimental assay can confirm their predictions. In particular, they mention that the availability of protein domain annotations in the InterPro database (3052 signatures as of June 2000) was a limiting factor, as only 50% of interacting proteins had sequence annotations. This may have reflected the incompleteness of the database, but may also suggest that protein sequences could have conserved linear regions of different types which are not identified by InterPro but which may still participate in mediating protein-protein interactions.

Bock and Gough [11] developed a method which represents proteins by the physicochemical properties (charge, hydrophobicity and others) of each individual residue and trains a machine learning model (support vector machine or SVM) to learn

the features which can predict interactions. Their training set is composed of a positive set of known true interactions and a negative set of randomized protein sequences which are unlikely to interact. Since most machine learning models, including SVM, require a fixed number of input features, one challenge when applying these methods to interaction prediction is mapping the widely-varying lengths of proteins into a fixed number of features. Bock and Gough make a point of showing the wide variety of sequence lengths in their data set, but few details are provided on their solution to the crucial fixed feature length problem. Also, the final number of SVM features is not specified. Nevertheless, they claim good performance (80% accuracy) on a test set of approximately 2,200 positives and 2,200 negatives. Assuming the error is evenly distributed between positive and negative instances, this corresponds to a sensitivity of 90% and a specificity of 90%. Even a relatively high specificity of 90%, as mentioned above, is not sufficient for all-to-all screens and would incur a large number of false positive errors.

Martin et al. [35] drew from Sprinzak's and Bock's works to create a method based on co-occurence of a type of sequence signatures in interacting pairs, but used an SVM model to learn the sets of signatures which are most informative for interaction prediction, rather than just counting occurrences. They represent each variable-length protein as a set of signatures which accounts for the global sequence composition of the protein and also for the compositions of the neighbourhoods of each residue along the protein, a representation inspired by the signature descriptor, which has been successful in various chemical informatics applications. Because the set of amino acids is finite, the set of possible amino acids and neighbours is also finite, leading to a fixed length feature vector for each protein sequence. Using this representation, they construct a kernel which

allows the SVM to compare pairs of signatures found in the query proteins to pairs of signatures found in training instances. They evaluate their classifier separately on several previously published data sets, by performing 10-fold cross-validation on each data set separately, claiming improved performance in direct comparisons to previous methods on the same data sets, including those of Sprinzak and Bock. However, basic information such as the numbers of positive and negatives instances in their data sets is omitted, and their performance measurements of other methods do not correspond with the reports in the various authors' original publications (for example, Sprinzak reports 97% sensitivity for their own work, while Martin reports 50% on the same data set). From a theoretical standpoint, their approach to modelling protein sequences is intriguing, because it is purely sequence-based: it does not rely on sequence annotations of any kind (such as domains), nor does it require mapping through amino acid physicochemical property scales. It also provides a straightforward and reusable approach to representing variable-length amino acid sequences in a fixed-dimensional space suitable for input to any other machine learning technique.

Shen et al. [36] adopt a similar approach for sequence representation by first grouping amino acid types by physical characteristics into 7 groups, then encoding a protein sequence as the frequency of occurrence in sequence of the $7 \times 7 \times 7 = 343$ possible 3-tuples of these residue types. This is similar to Martin's signature molecular descriptor, but also takes into account neighbour ordering. Pairs of interacting proteins are represented by concatenating their individual feature vectors, and they define a symmetric exponential SVM kernel for the distance between two pairs of proteins. The symmetric kernel makes their method insensitive to protein pair ordering. Their training

sets consist of 32,486 protein pairs (equal numbers of positives and negatives) and their test sets contain 400 protein pairs (the authors do not specify but seem to imply that their test set is also balanced). On these sets, they claim approximately 85% sensitivity and 84% precision. Given that their test set contains equal numbers of positives and negatives, this is equivalent to 81% specificity, which would result in a very large number of false positives in an all-to-all screen. In fact, when they validate their method by predicting previously documented protein networks, they only probe protein pairs among the list of proteins already known to be involved in the network. This increases the expected probability of interaction of those protein pairs and is not representative of the overall expected rate of interaction between all possible pairs. Nevertheless, their method further confirms that the SVM as a machine learning tool performs well in interaction prediction, and suggests that physicochemical properties of residues are a strong candidate features for PPI predictors because reducing the residue types into groups of similar physicochemical characteristics preserved classifier performance.

Guo at al. [37] encode residues using 7 physicochemical properties, but then take the auto covariance of these values at lags of up to 30 positions, to use as SVM features. Auto covariance and cross covariance features have previously been used successfully as features for DNA sequences [38] and in time series analysis [39]. They report 5-fold cross-validated results on 2,378 positives and 2,378 negatives. They explored several methods for generating the negative set: (a) pairs of proteins which are not co-localized, (b) pairs of proteins taken randomly from the positive set, except those known to interact, and (c) pairs of artificial protein sequences, mimicking distributions of single residues, tuples or triplets of residues from the real proteome. As expected, and as previously

suggested by Ben-Hur et al. [40], performance estimates for the non-co-localized negatives were biased toward better performance, because the classifier tends to learn to distinguish co-localized from non co-localized pairs rather than learning PPI prediction. Performance was also quite high for the residue shuffling sets, but these do not account for the complex characteristics of true protein sequences, such as protein domains or the 3D structure determined by true sequences, thereby again introducing unintentional differences between the positive and negative set. Their lowest performance was on the data set of pairs chosen from the set of proteins with some known interactions, where they report 58% accuracy while Shen has previously achieved 84% accuracy for that set. Guo's classifier achieves 42% sensitivity and 63% precision (equivalent 75% specificity) in this case, which are again insufficient for proteome-wide assays.

The method of Pitre et al. (S Pitre et al., 2008), the Protein-protein Interaction Prediction Engine (PIPE), is based on the early ideas in sequence-based PPI prediction of Sprinzak & Margalit, the co-occurrence of interacting regions in protein pairs known to interact. However, they extend the definition of "interacting regions" to arbitrary sequence windows and measure similarity between two sequence windows using the PAM similarity matrix [41]. By counting the number of known interacting pairs with similar windows in both query proteins, this method generates a landscape of window co-occurrence for each pair of windows, one window from each query protein (further explained in section 2.3.1). This landscape is reduced to a single score through an aggregation function such as median, mean, or maximum and a threshold is applied to generate a binary prediction. In their latest publication [17], they achieve a specificity of 99.95% and a sensitivity of 14.6% on a data set of 1,274 positive and 100,000 negative

interactions in *S. cerevisiae*, and by varying the threshold can achieve a specificity of 90% with 55% sensitivity. They apply their 99.95% specificity method to an all-to-all interaction screen in *S. cerevisiae*, identify 14,000 novel interactions, and experimentally validate some of their predictions. Despite promising results in terms of very high specificity, PIPE still only achieves 14.6% sensitivity, which is not sufficient to substantially enrich the global interaction network of *S. cerevisiae* already obtained by experimental methods. The authors conclude that improvements to PIPE are possible, perhaps by considering additional protein data such as localization or sequence annotations.

Park undertook an objective comparative analysis of the methods of Martin et al., Shen et al., Pitre et al., and Guo et al. [15]. He re-implemented the methods and applied them to the same datasets of ~5,800 *S. cerevisiae* and 34,862 *H. sapiens* protein pairs in 4-fold cross-validation testing. He found that the method by Pitre was the single best method, both in terms of precision at 20% recall (33% precision for PIPE, compared to 11%, 4% and 5% for Martin, Shen and Guo, respectively) and in terms of the methods' sensitivity at high specificity, as seen in the classifiers' ROC curves (see Figure 1 in Park's publication). Park carried out his experiment on test sets with 10:1 negative to positive ratio and 100:1 negative to positive ratio, confirming that lower negative to positive ratio artificially increases estimates of precision, while the ROC curves (in terms of sensitivity and specificity remain unchanged with respect to the ratio. Park also constructed a consensus classifier between the 4 methods, which slightly outperformed PIPE. This suggests that while PIPE is the strongest individual classifier, the other methods were still able to contribute improved predictions in certain cases, motivating

further work into improving PIPE. Park also comments that tests with higher ratios of negative to positives would have resulted in computationally prohibitive test sets sizes, which suggests that the issue of computational performance in protein-protein prediction has not yet been addressed, considering that proteome-wide screens consist of orders of magnitude more pairs than Park processed during his tests.

The methods reviewed in this section are summarized in Table 1. PIPE surfaces as the leader in this field and, while its performance is very good and has been verified independently, there are still improvements to be made in terms of sensitivity and perhaps computational efficiency to increase its usefulness as a tool for exploring organism-wide protein interaction networks. The next section provides a detailed description of PIPE, which enables a more detailed analysis of its performance and proposals for improvements in Chapters 3 and 4.

Method	Protein representation	Machine learning model	Performance	Reviewed by Park
Sprinzak & Margalit, 2001	Sequence-signatures from InterPro domain database	Log-odds of signature pair occurrence compared to random	97% sensitivity, unknown specificity	
Bock & Gough, 2001	Residue physicochemical properties, with unspecified mapping to fixed-length feature vectors	SVM	80% accuracy (equivalent 90% sensitivity, 90% specificity)	
Martin, 2005	Signature molecular descriptor (frequency of occurrence of residue and its neighbours)	SVM	70-80% accuracy (90% sensitivity, 90% specificity)	Yes (rank 2)
Shen, 2007	Conjoint triad (frequency of occurrence of 3-tuples of residue groups)	SVM	85% sensitivity, 82% precision (81% specificity)	Yes (rank 3)
Guo, 2008	Auto-covariance of physicochemical residue properties at lags of up to 30 positions	SVM	42% sensitivity, 75% specificity	Yes (rank 4)
Pitre, 2008 (PIPE)	Sequence windows compared using the PAM similarity matrix	Co-occurrence of window pairs in known interacting pairs	14.6% sensitivity at 99.95% specificity, or 55% sensitivity at 90% specificity	Yes (rank 1)
Park, 2009	Varies	Consensus of previous 4 methods	60% sensitivity at 90% specificity	N/A

Table 1: Summary of six leading protein-protein prediction methods

2.3.1 Protein-Protein Interaction Prediction Engine (PIPE)

The Protein-protein Interaction Prediction Engine (PIPE) is a method developed at Carleton University by the bioinformatics research group, as a collaboration between faculty and graduate students from Computer Science, Biology and Engineering. Dr. Sylvain Pitre is the primary author of the method and wrote the initial implementation of the method [16], but the model and implementation have since been revised to improve classification and computational performance [17]. The work described here was done in

collaboration with the Carleton University bioinformatics research group, and the latest source code of PIPE was obtained directly from the main author, Sylvain Pitre. Modifications which improved the performance of PIPE were contributed back to the bioinformatics research group.

PIPE predicts protein interactions from protein sequence alone [16] [17]. Given two protein sequences and a database of protein pairs known to interact, PIPE outputs a pair score for the potential interaction between the two query proteins. PIPE is thus a binary classifier where the input space is unordered pairs of proteins, the training data are the known interactions (in the same sense as a nearest neighbour classifier uses training instances), and the output is a score which can be used to rank predictions or, given a threshold, to make a binary decision. A high-level system overview is shown in Figure 3.

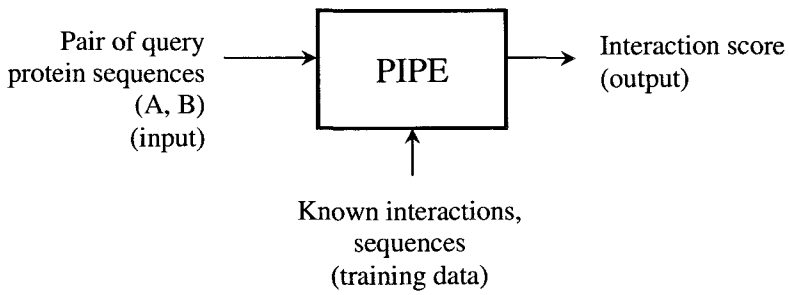


Figure 3: PIPE high-level system overview

The core hypothesis of PIPE is that if query protein A has sequence homology with protein A′, query protein B has sequence homology with protein B′, and proteins A′ and B′ have been shown to interact experimentally, then we can expect that A and B would also interact in a similar way. This resembles a nearest neighbour classifier in the space of protein pairs. The PIPE hypothesis is based on the nature of protein structure. A protein's amino acid sequence determines its three-dimensional structure, while the physical and chemical properties of the amino acids at the surface of the protein

determine its interactions with other proteins. Thus, pairs of amino acid sequences which are often found in interacting protein pairs known are likely to represent complementary three-dimensional structures. When present in a new pair of proteins for which only the sequences are known, the re-occurring sequence pairs suggest an interaction between the two new proteins. This is similar to Sprinzak and Margalit's argument [34], but extended so that the unit of re-occurrence is arbitrary sequences rather than only protein domains.

PIPE finds homology between proteins by comparing 20-amino acid windows using the PAM amino acid similarity matrix [41]. Dayhoff derived this matrix from the likelihoods of amino acid mutations observed in a set of evolutionarily-related proteins which were thought to have similar function. This matrix provides a similarity score for each pair of amino acids. The computation of the PAM matrix and the assumptions of the underlying statistical model of evolution are discussed in detail by Dayhoff [41]. The similarity of two amino acid windows of length 20 is measured as the sum of the PAM scores (specifically, PAM120 scores) of amino acids at corresponding positions in the windows. Windows which have a large degree of similarity achieve higher PAM scores and are more likely to be structurally and functionally related. The score threshold is chosen by finding the threshold which achieves a very low probability (10^{-6}) of accepting a match between two random windows. PIPE uses a threshold of 35 for *S. cerevisiae* and 40 for *H. sapiens* and *M. musculus*.

One implicit assumption when using sequence windows is that protein regions which mediate interaction are contiguous in the amino acid sequence of the protein. This ignores long-range effects where amino acids distant in the sequence of the protein become physically clustered when the protein assumes its folded three-dimensional

structure. It is hard to estimate how many PPIs are mediated by these long-range effects rather than local interactions between short linear stretches of amino acids. Nevertheless, linear peptides have been previously shown to mediate some PPIs [42], and PIPE itself has been shown to achieve good performance compared to other sequence-based PPI predictors (section 2.3).

For each window of amino acids in query protein A, PIPE finds proteins within the database of known interactions which have at least one window which is similar to the query window. The same is done for each window in query protein B. Pairs of proteins with a high degree of sequence similarity to each of A and B which are also found in the list of known interacting pairs provide evidence for interaction of the two original query windows. The number of co-occurrences of such homologous windows is the number of "hits" for the pair of windows from the query proteins. This is calculated for all pairs of query windows, resulting in a matrix of values (also referred to as a PIPE landscape) which represents the local likelihood of interaction between each pair of regions from the two query proteins (Figure 4). In the example shown, the region around YPL240C amino acid 220 appears to interact with large portions of YAL005C, but in particular with the region around amino acid 500. Higher overall matrix values suggest an interaction between the two proteins as a whole.

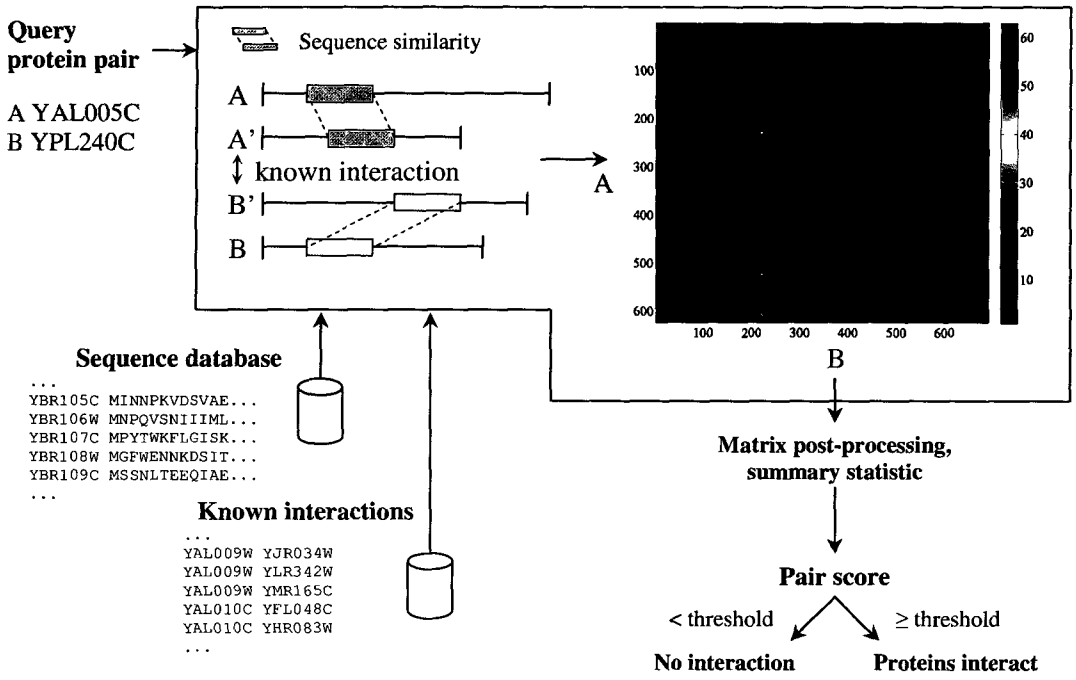
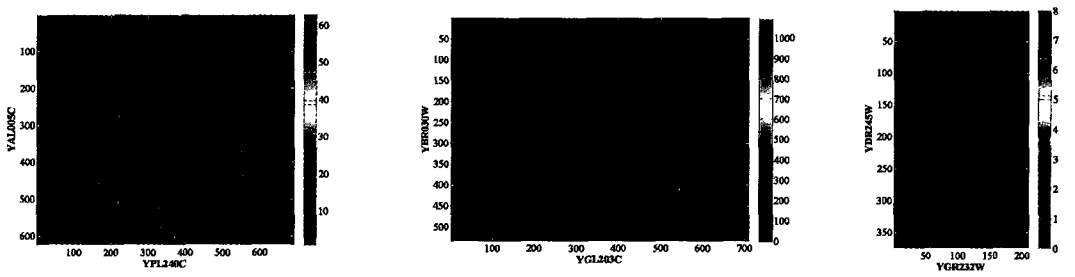


Figure 4: Internal block diagram of PIPE

The PIPE matrix exhibits several important features. First, some matrices contain long, thin ridges which are thought to represent false hits since it is unlikely that one protein window interacts with an entire partner protein (Figure 5a). PIPE applies a 3x3 median filter to remove these regions. For computational efficiency, this filter is implemented by simply counting non-zero cell values among the 9 values within the filter's coverage, so height information is effectively discarded. PIPE matrices also have compact regions of high matrix values which are thought to represent the binding sites between the two proteins (Figure 5b). On the other hand, some matrices have only background noise and these represent a lack of interaction between the two proteins (Figure 5c). To reduce the matrix to a single value, PIPE calculates the mean of the matrix values.



(a) (b) (c)

Figure 5: Sample PIPE matrices showing some common matrix features. (a) a thin long ridge (false hits) (b) a compact region of high matrix values (binding sites) and (c) background noise (no interaction)

When applying PIPE to a query pair which is also included in the list of known interactions, the PIPE matrix automatically assumes a baseline value of 1, because each query protein has homology with itself, and the query pair is a known interaction. Because of this, PIPE is inherently biased toward higher scores for pairs which appear in the known interactions list, which biases performance estimates on the training set. Therefore, it is particularly important to apply some form of cross-validation to reliably estimate the model's performance on unseen protein pairs. In particular, leave-one-out cross-validation (LOOCV) is well-suited for this purpose because excluding one known pair from the PIPE algorithm is computationally inexpensive, and it provides a large sample of instances for estimating the classifier's performance, which allows for very accurate estimates. For PIPE in particular, there are several ways of excluding training data for each test pair, with varying levels of stringency:

1. Exclude only the test pair (A, B) from the known interactions. This is the traditional interpretation of "leave-one-out" and the least stringent as it excludes only one known interaction from the training data, thus not very strongly affecting PIPE matrices.

2. Exclude all protein pairs of the form (A, *) or (*, A). This is a more stringent approach, as it excludes all of A's known interactions from the training data. PIPE must rely on homology between A and *other* proteins to establish evidence for interactions with A.
3. Exclude all protein pairs of the form (A, *), (*, A), (B, *) or (*, B). This is the most stringent LOOCV case, because PIPE does not directly benefit from any of A's or B's interactions. Both A and B must have homology with known interactors, other than themselves.

All test results presented in this thesis were obtained using the last, most stringent method. This provides the most confidence that the method and modifications presented here are not simply over-fitting to the training data and that LOOCV performance is a reliable estimate of performance on unseen data.

3 Computational Acceleration of PIPE using Improved Data Structures and a Space-Time Trade-off

Many different types of protein-protein interaction predictors have been proposed based on different protein sequence representations or machine learning models (see section 2.3). Authors frequently report their method's classification performance, suggesting that they trained and tested their models on actual data sets of protein pairs. An implicit assumption is that the training and testing algorithms were computationally efficient enough to execute in a reasonable amounts of time. The abundance of methods suggests that developing and validating PPI classifiers at the scales suggested in existing publications is achievable using widely-available computer hardware. On the other hand, this issue of computational performance of PPI classifiers has never been addressed explicitly.

At precisely what scale does this assumption of reasonable training and testing times remain valid? For most machine learning methods, the training and testing times increase with the number of instances. Studies commonly cite application of their methods on databases of known interactions of 10,000 protein pairs in *S. cerevisiae* [37] and 16,000 protein pairs in *H. sapiens* [36]. However, real PPI datasets are substantially larger than this. For example, HPRD, a large repository of *H. sapiens* protein-protein interactions, contained 39,167 protein pairs in its 2009 update [43]. Furthermore, these sizes are expected to increase as more interactions are experimentally discovered. Also, larger numbers of protein pairs known to interact means more labelled training and test instances. Larger training sets enable better training of existing models (lower model

variance) and training of higher-order models with less danger of overfitting, while larger test sets increase the accuracy of classifier performance estimates. In *H. sapiens*, the estimated total number of interactions is 650,000 [7]. Therefore, databases of known interactions in *H. sapiens* can be expected to increase in size by at least one order of magnitude.

Classifier testing also puts pressure on the computational performance of PPI classifier implementations. In order to measure the predictive performance of a classifier, we must execute the classifier implementation for every test instance. This means that the testing time also increases with the number of test instances. Furthermore, because we wish to achieve very high specificity (above 99.9%, as discussed in section 2.1), it is necessary to have a very large pool of negative test instances (on the order of 100,000) in order to obtain accurate estimates of specificity. Existing methods often side-step this issue and perform model testing on sets deficient in negative instances (as discussed in sections 2.1 and 2.2, and by Park [15]), resulting in potentially biased classification performance estimates depending on the performance measures which the authors chose to report (precision and accuracy are biased, as explained in section 2.1). Ideally, one would use a test set with a ratio of positives to negatives that approximates the ratio expected to be found in the cell (1:370 in *H. sapiens*). If one was to use the current data sets of known positive interactions and add to those a number of negative protein pairs in the correct ratio, the total test set size would be more than two orders of magnitude larger than current test sets, and testing time would increase accordingly. Even with optimized classification algorithms, some authors are unable to process data sets more strongly biased towards negatives than a positive to negative ratio of 1:15 [44]. Tweaking the

model and performing multiple iterations of training and testing, which usually results in better models, simply becomes intractable.

Finally, one of the main goals of PPI classifier design is the application of the trained models to organism-wide interaction screens, directly addressing the need of systems biologists for whole-organism protein interaction networks. This interaction network can be obtained from an all-to-all screen, which applies the classifier to all (unordered) pairs of proteins in an organism, including self-interactions, and which would need to consider $\frac{numProteins^2+numProteins}{2}$ protein pairs. Assuming 6,000 proteins in *S. cerevisiae* and 20,000 in *H. sapiens*, the classifier would have to be applied to approximately 18 million and 400 million protein pairs, respectively. This number of protein pairs is larger than the size of test sets in previously published methods by 3 orders of magnitude in *S. cerevisiae* and 4 orders of magnitude in *H. sapiens*. Note also that this number increases quadratically with the number of proteins in the organism. Only the PIPE method has attempted an all-to-all screen in *S. cerevisiae*, and it required 1,000 hours of continuous computation on a large computational cluster [16]. This was later reduced in an improved implementation of the algorithm to 48 hours of computation on 76 processors [17]. An all-to-all screen in *H. sapiens* required a month of continuous computation on two clusters (unpublished results; at the time, one of the clusters was ranked in the top 5 on the Top500 list of supercomputers [45]). In all three cases, the screens were out of reach for researchers without access to large computational clusters.

In the following sections we describe the most up to date implementation of the PIPE algorithm [17], obtained as C source code from Sylvain Pitre. This is the only available implementation of PIPE and is maintained by the bioinformatics research group

at Carleton University. We describe PIPE in detail and discuss the algorithmic complexity and practical runtimes of different parts of the algorithm depending on the characteristics of the proteome on which the algorithm is being used. We identify several aspects of the implementation of the algorithm which could be improved and propose modified algorithms with our improvements. This new implementation achieves up to 80 times speedup (14.5 on average) compared to the original implementation for classifier test runs. This facilitated the development and validation of the classification model improvements described in the following chapter (Chapter 4). We estimate that an all-to-all screen in *H. sapiens* would complete approximately 14.5 times faster, making it possible to attempt such a screen in less time and using less computational resources than before.

The analysis and changes proposed in this section sometimes refer to the algorithm, and sometimes to the implementation, of PIPE. Algorithm changes primarily consist in alternate ways of computing the same result, but in a way which trades reduced time complexity for another computational resource such as memory utilization. Implementation changes primarily modify the memory layout and data representation used, without significantly changing the sequence of operations or data processing involved. This is sometimes accompanied by small changes in the algorithm, such as additional initialization required at different points, different methods of iterating over a list, or a constraint on the applicable algorithms for a particular operation. In some cases, the distinction between algorithm and implementation changes is blurred. However, both types of changes contribute improvements in execution speed, and together can improve the overall speed of PIPE and its utility as a PPI predictor.

3.1 PIPE Algorithm

When processing a given protein pair, PIPE examines each sequence window in query protein A for similarity to other proteins A' in the organism's proteome (refer to Figure 4). When a match for the current window in query protein A is found in another protein A', PIPE looks at each known interaction partner of A' for a potential protein B' which contains a window matching the current window in the second query protein B. This process is repeated for each pair of window sequences from the query proteins A and B. The result is a matrix of co-occurrence of pairs of sequence windows from A and B, among proteins which are known to interact.

In the initial implementation of PIPE [16], the sequence window comparisons were performed on the fly. For each sequence window in A and each sequence window in B, PIPE compared these windows to every other window in the proteome. This was the most computationally intensive part of the algorithm. Because both query proteins and proteins in the known interactions list both originate from the known proteome, these sequence comparisons were always performed between two sequence windows from the proteome and in practice the same window comparisons were often re-computed. In the revised implementation [17], PIPE first precomputes all window comparisons and stores the results on disk, which accelerates the execution of the main algorithm. For every sequence position i in every protein A in the proteome, the precomputation step compares the reference window A_i to all proteins in the database and outputs the list of proteins which contain a window which is similar to A_i. These similarity lists are grouped and written to disk sequentially, one file for each protein A. This optimization enables the outer loop of PIPE (iteration over every sequence window in protein A) to

only require reading the similarity lists from disk and iterating over them, rather than comparing the sequence windows to all other sequence windows in the proteome on the fly.

A pseudocode representation of the PIPE algorithm is shown in Figure 6. The algorithm takes as inputs the query protein pair `A` and `B`, the precomputed sequence window comparisons described previously and a mapping from each protein to its list of known interaction partners. Proteins are represented as unsigned 16-bit integers, which are the indices of the proteins in the list of protein sequences. This places a limit of 65,536 on the total number of proteins, but this has not been an issue in practice. The list of known interacting pair is converted to a mapping by grouping the list by the first protein in each pair. Thus, the key of the mapping is any given protein in the proteome, and the value of the mapping is a list of proteins which are the key protein's partners. Pairs in the known interactions list are taken as symmetric, that is, if (`A`, `B`) is present in the original list, then `B` is in `A`'s known partners list, and `A` is in `B`'s known partners list. The "mapping" representation ensures constant-time access to the list of known interactions for any arbitrary protein `X`.

```
     Inputs: i) query protein pair (A, B),
             ii) precomputed sequence window comparisons for every
                 protein:
               Protein x:
                 Window i -> (list of proteins with similar windows)
             iii) known interacting pairs, indexed by one of the
                  proteins:
               Protein x -> (list of known interaction partners)
     Outputs: matrix of co-occurrence H,
                dimensions (length(A), length(B))

  1  fill output matrix H (dimensions length(A) by length(B)) with 0

  2  for each sequence window A_i in A
  3    {A'} = list of proteins with windows similar to
              protein A's window A_i

  4    for each A' in {A'}
         // A' is a protein similar to A
  5      {B*} = list of proteins known to interact with A'

  6      for each B* in {B*}
           // B* is known to interact with a protein similar to A,
           // but we must also check for similarity to B'.
  7        for each sequence window B_j in B
  8          {B'} = list of proteins with windows similar to
                    protein B's window B_j

  9          for each B' in {B'}
 10            if B' == B*
                 // Window A_i is similar to a window in A', which
                 // interacts with B*. B* is the same protein as B',
                 // which contains a window similar to B_j. Therefore,
                 // it's possible that windows A_i and B_j interact.
                 // Refer to Figure 4.

                 // Increment the matrix cell corresponding to the
                 // current window in A (i) and the current window in B
 11              // (j).
                 H_i,j += 1
 12
 13            end if
 14          end for
 15        end for
 16      end for
 17    end for
     end for
 18
     return H
```

Figure 6: PIPE algorithm from Pitre et al. 2008 [17]

Each of the nested loops in Figure 6 directly corresponds to the steps of the PIPE algorithm described at the beginning of this section. The `if` statement (line 10 in Figure 6) is executed a number of times which depends directly on the number of iterations of

the enclosing loops, and this statement is taken as the unit of execution. The cost of the increment operation inside the `if` statement is weighted by the probability of the branch being taken, so we consider it negligible. When a section of code is contained within a loop, we calculate the total cost of the loop as the cost of the loop body multiplied by the average expected number of iterations. Here, the expected number of iterations for a given loop is a function of the proteome from which the query proteins and input databases are derived, as described below. This multiplication does not give an exact result because the numbers of iterations in different loop levels are somewhat correlated, which means that the result is a biased estimate of the total number of iterations. Nevertheless, this is sufficient for a cursory analysis of the algorithm. The effect of this correlation is discussed in detail later, at the end of section 3.3.

The parameters of the proteome which influence algorithm runtime and memory requirements are listed in Table 2 for three organisms: *S. cerevisiae*, *M. musculus*, and *H. sapiens*. The average values are used for calculating average time complexity, while maximum values are used for calculating the worst-case space complexity (memory requirements). For the time complexity of the algorithm, we can associate each nested loop with one of these parameters (Table 3). The total time complexity of the entire algorithm, for a particular protein pair (`A`, `B`), is O(lengthA × avgSimHits × lengthB × avgNeighbours × avgSimHits) operations. The algorithm does not use any additional memory aside from the identifiers of the input proteins, the protein similarity database, the interaction pair database and the output matrix.

Parameter	Symbol	Value in *S. cerevisiae*	Value in *M. musculus*	Value in *H. sapiens*
Length of a protein (number of amino acids)				
For a particular protein	lengthA, lengthB			
Over all proteins				
Minimum	minLength	16	5	30
Average	avgLength	449.7	555.6	543.2
Maximum	maxLength	4,910	35,213	34,350
Number of proteins				
Entire proteome	numProteins	6,716	16,354	22,513
Number of amino acids (proteome size)				
All organism proteins	proteomeSize	3.0×10^6	9.1×10^6	12.2×10^6
Number of known interactions				
	knownPairs	43,591	3,580	41,678
Number of pairs in an all-to-all screen, including self-interactions = $\frac{numProteins^2+numProteins}{2}$				
	allPairs	22.6×10^6	133.7×10^6	253.4×10^6
For a given sequence window, number of proteins containing a window similar to the given window (similarity hits)				
For a particular sequence window	$simHitsA_i$, $simHitsB_i$			
Over all sequence windows				
Minimum	minSimHits	0	0	0
Average	avgSimHits	7.9	8.4	21.3
Maximum	maxSimHits	550	1,187	1,680
For a given proteins, number of its known interaction partners				
For a particular protein	neighboursA'			
Over all proteins				
Minimum	minNeighbours	0	0	0
Average	avgNeighbours	12.83	0.43	3.60
Maximum	maxNeighbours	334	257	264

Table 2: Characteristics of the protein sequences and protein interactions in *S. cerevisiae*, *M. musculus* and *H. sapiens*

Line number in Figure 6	Loop description	Approximate number of iterations
2	Every sequence window in A	lengthA
4	Every protein with a window similar to A_i	avgSimHits
6	Every protein known to interact with A'	avgNeighbours
7	Every sequence window in B	lengthB
9	Every protein with a window similar to B_j	avgSimHits

Table 3: Nested loops in the PIPE algorithm and their associated approximate numbers of iterations

Because PIPE mainly iterates over lists of integers, with very little arithmetic or logic operations, the algorithm's performance is heavily constrained by how efficiently it uses the memory subsystem of the hardware on which it is executing. The on-disk and in-memory representation of the inputs to the algorithm can have a large impact on its memory access patterns, and therefore on its performance. In particular, the cost of accessing main memory is the dominant factor and the data structures chosen must be amenable to caching inside the CPU. In general, storing more frequently-used data in cache leads to better cache efficiency, higher cache hit rate, and better amortizes the cost of performing an external memory access to main memory.

3.2 Input Data Representation

3.2.1 Similarity Lists of Protein A

As described in section 3.1, the window comparison precomputation generates a file for each protein in the organism. Each file contains unsigned 16-bit integers which encode the results of the sequence window comparisons (Figure 7). Each of the integers is stored as its raw 16-bit binary representation. For protein `A`, the results are stored in a file named "`A`" which contains: the number of amino acids in `A` (lengthA) and, for each sequence window A_i, the number of proteins which contain a window similar to A_i and

the actual list of protein identifiers. This naturally leads to the loops on lines 2 and 3 in Figure 6 as simply sequentially reading and parsing integers from this file.

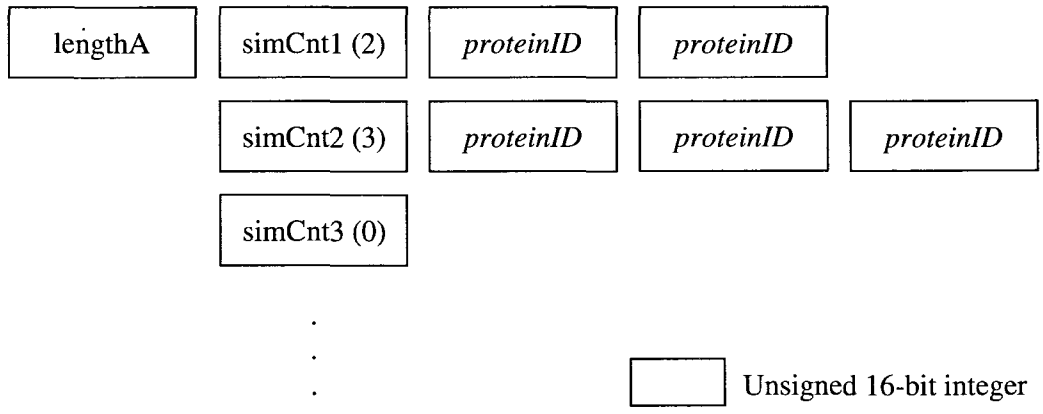


Figure 7: On-disk format of the precomputed similarity files

The current implementation uses the C library function "fread" to read from the similarity files. This function is called for every integer to be read from the similarity file, which means that it is called on average avgLength × avgSimHits times for reading the similarity file of protein `A` for every protein pair. Function calls have associated overhead and minimizing the number of function calls, particularly in a very commonly-used area of a program, is likely to lead to performance improvements. Therefore, we propose that the entire similarity file be read into an in-memory array at once (a single call to fread), and that this retrieval of the protein lists be implemented as a pointer sliding along the array. The loop on line 2 is then implemented as a pointer dereference (much less expensive than a function call), and the set `{A'}` is also accessed by indexing into an in-memory array of protein identifiers. The similarity files are at most 186 kilobytes in *S. cerevisiae* and 1.2 megabytes in *H. sapiens*, which is well within the range of typical memory capacities, and can even fit in the CPU caches of many modern processors.

An alternative method of accessing the similarity files would be to memory-map these files into the virtual address space of the PIPE process. Like the in-memory buffer, this saves one function call per element access compared to the original implementation. However, unlike the in-memory buffer which keeps a private copy of the similarity data for each PIPE instance, the memory mapping method keeps only one copy of the similarity data in physical memory (in the operating system cache), and all PIPE instances access the same physical memory, reducing the overall memory usage for multiple instances of PIPE. This would also have the advantage of more efficiently using the CPU cache, by not duplicating data in physical memory, thus not caching duplicate data in the CPU. However, these may not be relevant factors because the implementation is not bound by available memory, and the likelihood of two PIPE instances processing the same proteins simultaneously and benefiting from the cache sharing is fairly low, especially as the total number of pairs grows. While we have not attempted to use memory-mapping for similarity files, we expect that the practical performance gains may not be noticeable.

3.2.2 Similarity Lists of Protein B

The reading and processing of the similarity data for protein `B` is also a potential bottleneck in the algorithm. Because the iteration over all sequence windows in `B` is nested much deeper in the algorithm, it is executed many more times. On average, the loop over all proteins in B_j's similarity list (line 9) is executed avgLength × avgSimHits × avgNeighbours × avgLength times.

The current implementation reads the similarity file of `B` only once, at the beginning of the algorithm, using the same "fread" function call, and pre-processes all

similarity lists of B into an array of singly linked lists (Figure 8). The loop over all sequence windows B_j (line 7) is implemented by indexing into this array, and the loop over each protein in B_j's similarity list (line 9) is implemented by following the node pointers in the linked list data structure (Figure 8).

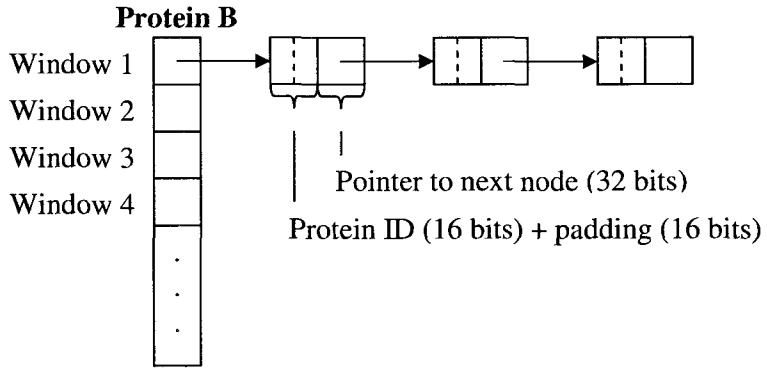


Figure 8: In-memory representation of the protein B similarity lists

This pre-processing step of the similarity lists of B is not necessary. In fact, this representation has 75% overhead because only 25% (16 out of 64 bits) of the node structures contain the actual protein identifiers which are useful to the algorithm's processing. This means that only 25% of the CPU cache space occupied by this data structure contains useful data. Furthermore, the node structures are allocated dynamically using "malloc", which makes no guarantee about the spatial locality of the returned memory regions. Ensuring spatial locality of the node structures would speed up sequential iteration of the nodes by taking advantage of prefeteching logic in the CPU cache, which would fetch multiple nodes from main memory at once.

We propose that the similarity file be read entirely into memory, as proposed in section 3.2.1 for protein A. The loops over each window in protein B (line 7) and each protein in the similarity list for B_j (line 9) are implemented as a pointer sliding along the

array, referencing the appropriate sub-sequence of integers when the set {B'} is required. This representation has very good spatial locality because it iterates sequentially over adjacent locations in memory. When the first element of the similarity set {B'} is accessed and brought into CPU cache from main memory, subsequent elements are also prefetched and ready for the next iteration of the loop without accessing main memory again. Furthermore, the CPU cache is used very efficiently, because by storing the protein IDs sequentially and using the minimum number of bits (16-bits for proteomes of less than 65,536 proteins), all of the CPU cache is filled with useful data, allowing 4 times more protein identifiers to be stored in cache than the original implementation, for the same physical cache size.

3.2.3 Known Interactions Lists

On line 5, the algorithm retrieves a list of proteins {B*} known to interact with A'. The same linked list representation was also used for the storage of the lists of known partners for each protein. The loop over each partner (line 6) is implemented by following the "next" pointer of each linked list node. The same CPU cache efficiency argument applies as in the case of the similarity lists of protein B (section 3.2.2) and this loop would also benefit from the plain array representation of the protein identifier list.

However, for the known interactions list, we must be careful to preserve constant time random access of the known interactions list for any arbitrary protein. This is used on line 5 to access the known interactions of A', where A' is not known ahead of time and is not iterated sequentially. To preserve constant time lookup, we still store the known interactions lists contiguously in memory, but we pad each list to the maximum length (334 in the worst case for *S. cerevisiae*, so this is not memory prohibitive).

Accessing the known interactions list for an arbitrary protein k then consists of computing the offset of the list, k × maxNeighbours. This has some main memory overhead due to the unused slots in lists of proteins with few known interactions. However, more importantly, this does not pollute the CPU cache, because those memory locations are never accessed, and preserves near 100% efficiency of the cache as in section 3.2.2.

3.2.4 Reloading of Protein B Similarity Data

Furthermore, because the algorithm is often used to process a large number of protein pairs, it is worth preserving in memory as much data as possible from one pair to another. In particular, the original implementation reloaded the protein `B` similarity file for every protein pair. We propose that a check be added that avoids reloading the similarity file when protein `B` doesn't change from one protein pair to another. The benefit of this optimization can be maximized, when performing all-vs-all experiments, by first ensuring that the protein list given to the PIPE algorithm is sorted by protein `B` before executing PIPE. This pre-sorting does not affect the results of the algorithm but provides an advantage in total runtime for all input protein pairs. Additionally, because subsequent optimizations, described in section 3.3 below, will require some precomputation on the protein `B` similarity file, this optimization will become more beneficial as the results of the precomputation can be saved in memory for the next protein pair.

3.3 Space-Time Tradeoff

We now re-arrange the algorithm to facilitate further analysis. Notice that the list of sequence windows `B` (the set being iterated over in line 7 of Figure 6) is independent of the current `B*` (the current element of the iteration on line 6), which is a protein known to interact with `A'`. Therefore, changing the order of iteration of these two loops has no effect on the behaviour of the algorithm. By making this change, we can highlight a pair of nested loops which can be optimized. An equivalent, modified algorithm with these two loops swapped is shown in Figure 9.

```
     Inputs: i) query protein pair (A, B),
             ii) precomputed sequence window comparisons for every
                 protein:
               Protein x:
                 Window i -> (list of proteins with similar windows)
             iii) known interacting pairs, indexed by one of the
                  proteins:
               Protein x -> (list of known interaction partners)
     Outputs: matrix of co-occurrence H,
                dimensions (length(A), length(B))

  1  fill output matrix H with 0

  2  for each sequence window A_i in A
  3    {A'} = list of proteins with windows similar to
              protein A's window A_i

  4    for each A' in {A'}
         // A' is a protein similar to A
  5      {B*} = list of proteins known to interact with A'

  6      for each sequence window B_j in B
  7        {B'} = list of proteins with windows similar to
                  protein B's window B_j

           // {B*} is a list of partners of A' (similar to A_i).
           // {B'} is a list of proteins similar to B_j.
  8        for each B* in {B*}
  9          for each B' in {B'}
 10            if B' == B*
                 // Window A_i is similar to a window in A', which
                 // interacts with B*. B* is the same protein as B',
                 // which contains a window similar to B_j. Therefore,
                 // it's possible that windows A_i and B_j interact.
                 // Refer to Figure 4.

                 // Increment the matrix cell corresponding to the
                 // current window in A (i) and the current window in B
 11              // (j).
                 H_i,j += 1
 12
 13            end if
 14          end for
 15        end for
 16      end for
 17    end for
     end for
 18
     return H
```

Figure 9: PIPE algorithm with rearranged inner loops

Notice that the two innermost loops (lines 8 through 14) compute the number of proteins which are both partners of A' (in {B*}) and similar to B_j (in {B'}). This is effectively the size of the set intersection $|\{B^*\} \cap \{B'\}|$. The same method is used in the

original implementation from Figure 6 , where the loops are on lines 6 and 9. This nested loop method of calculating the set intersection requires on the order of avgNeighbours × avgSimHits iterations, and is a relatively inefficient method of calculating this intersection. By improving the implementation of this operation, we can dramatically improve the performance of the algorithm as a whole, particularly since this is in the innermost loops and is executed a large number of times (lengthA × avgSimHits × lengthB on average).

There are many potential approaches for computing set intersection efficiently [46] [47], with different time and space complexity, different requirements on the representation of the input sets, and with different practical runtimes depending on the relative sizes of the two sets.

Let the first set be `X` with `m` elements and the second set be `Y` with `n` elements. Currently, the outer loop iterates over each element of `X` and the inner loop is a set membership query in the `Y` set. The naive implementation is a linear search through the array, which requires O(n) steps. This is the current algorithm with overall complexity O(mn). However, if we sort the `Y` array ahead of time, then binary search can find an arbitrary element `x` in the `Y` array in O(log n) time. Thus, the set membership query in the `Y` set can be optimized from the linear search with O(n) complexity to binary search with O(log n) complexity. There are still `m` set membership queries, resulting in an overall set intersection with time complexity O(m log n), without any additional memory requirements (O(1) space complexity). This is already better than the original O(mn) complexity, particularly if we can select `Y` to be the larger of the two sets (`n > m`). This choice is arbitrary and does not change the results of the algorithm because set

intersection is a symmetric operation. Additionally, it should be noted that the {B*} and {B'} arrays are already sorted as a side-effect of the precomputation phase, so there is no additional penalty for ensuring this precondition is true.

We can use even more optimized algorithms if we impose additional restrictions on the input data. If we assume that both X and Y are sorted before the algorithm is executed, the set intersection can be computed using a binary merge algorithm. This is similar to the merge step in merge sort [48], where two sorted lists are merged by walking the two lists simultaneously. Instead of generating the combined output list as in merge sort, we simply keep a running count of the number of equal elements in each list. This method visits each list element once, so the time complexity of the set intersection is O(m + n), while the space complexity is still O(1). This is even better than O(m log n), and most likely could achieve several times speedup in practice compared to the original O(mn) algorithm.

If we tolerate an increase in space complexity, we can achieve O(m) time complexity. By precomputing an alternative representation of one of the sets, we can perform set membership queries in O(1) time. Specifically, we preprocess the Y set into a bit vector representation. This is a contiguous block of memory in which each bit represents whether a particular element from the universe is a member of the set: if the bit is 1, the element is in the set, otherwise, it is not. In this case, the universe is the set of all proteins of the organism under consideration. Until now, the sets have been represented as arrays of protein identifiers and the space complexity of the sets was O(n) where n was the size of the set. The bit vector representation has O(numProteins) space complexity because it requires a bit for each protein in the proteome. However, this representation

allows O(1) membership queries, by performing a single memory access to the correct location in the bit vector. The O(1) membership query is contained in a loop over all the elements of the X set, so in total we perform m membership queries. The overall time complexity of the set intersection is now O(m), which is a improvement from the original O(mn) complexity by a factor of n. However, the space complexity is O(|U|), where U is the universe of the Y set, which may be prohibitive depending on the size of the proteome and available hardware resources.

Table 4 summarizes the possible implementations of the set intersection and their characteristics.

Algorithm	Input requirements	Time complexity	Space complexity
Linear search	None	O(mn)	O(1)
Binary search	Y is sorted	O(m log n)	O(1)
Merge	X and Y are sorted	O(m + n)	O(1)
Bit vector	Y is represented as a bit vector	O(m)	O(|U|)

Table 4: Properties of four set intersection algorithms. These are candidates for the set intersection computed by the two inner loops of the original PIPE algorithm.

Because set intersection is a symmetric operation, we can choose which set ({B*} or {B′}) to assign to X and Y. There are two potentially competing goals: we wish to assign the largest set to Y, because that provides the best improvement of time complexity from the original implementation, by a factor of n, where n is the size of the Y set. On the other hand, for a complete execution of the algorithm for a given protein pair, the set intersection operates on several different {B*} and {B′} sets, depending on the current A′ or the current B_j, respectively. The total space requirement will depend on how many bit vectors must be precomputed and stored, so that reducing time complexity may drive an increase in practical memory requirements of the algorithm.

If we take Y = {B*}, the set of known interaction partners of A', then there is a set for each choice of A' (each protein in the proteome), for a total of numProteins sets. Each set requires numProteins bits so that the total memory requirement is numProteins^2 bits, or 5.6 megabytes in *S. cerevisiae*, 33.4 megabytes in *M. musculus* and 63.4 megabytes in *H. sapiens*. This choice of Y would result in a theoretical asymptotic speedup of avgNeighbours = 12.83 in *S. cerevisiae*, 0.43 in *M. musculus* and 3.60 in *H. sapiens*. In *M. musculus*, the bit vector algorithm can sometimes perform worse the linear search (original) algorithm: if Y = ∅, the bit vector algorithm always makes one memory access to probe the set for membership of the current x protein. The linear search algorithm need not probe the protein identifiers array after finding that the array is empty (Y = ∅).

On the other hand, if we choose Y = {B'}, the set of proteins with a window similar to B_j, there are lengthB such sets, one for each sequence window in B, and the total space requirement is lengthB × numProteins, for a given choice of B. Because lengthB varies widely, we must account for the worst case space requirements. For the case where B is the longest protein in the proteome, the space required is 4.1 megabytes in *S. cerevisiae*, 72.0 megabytes in *M. musculus* and 96.7 megabytes in *H. sapiens*. In the average case, however, the memory requirements are 377 kilobytes in *S. cerevisiae*, 1.1 megabytes in *M. musculus* and 1.5 megabytes in *H. sapiens*. The asymptotic speedup for this choice of Y would be avgSimHits = 7.9 in *S. cerevisiae*, 8.4 in *M. musculus* and 21.3 in *H. sapiens*.

Luckily, for these particular organisms, neither choice of Y is prohibitive in terms of space complexity, even in the worst case. Even when used in a multi-user computing

cluster environment, it's likely that each user process will be allowed to allocate at least 100 megabytes of memory. Therefore, our choice of the `Y` set is guided by optimization of the time complexity. For *S. cerevisiae*, it seem as though `Y = {B*}` offers the best speedup (12.83), while for *M. musculus* and *H. sapiens*, `Y = {B'}` is better, with speedups of 8.4 and 21.3, respectively.

We chose to set `Y = {B'}`, which favours *M. musculus* and *H. sapiens*. First, all-to-all screens in *M. musculus* and *H. sapiens* (133.7 and 253.4 million pairs) contain many more protein pairs than in *S. cerevisiae* (22.6 million pairs). This means that in terms of the time required for a full all-to-all screen, the greatest impact can be achieved when optimizing for these two cases. Additionally, *M. musculus* and *H. sapiens* are of more practical interest because results on these organisms are closer to potential biomedical applications. Furthermore, the fact that avgNeighbors > avgSimHits in *S. cerevisiae* is most likely the result of intense interest in studying the organism, driving up the number of known interactions and therefore avgNeighbours. We expect that most other organisms are much less studied than *S. cerevisiae*, and have much lower numbers of known interactions, and therefore low avgNeighbours values suggesting that our choice of `Y` is more widely-applicable.

One remaining concern is the scaling of the space complexity when applied to organisms with larger proteomes. For `Y = {B*}`, the space requirements reach 1 gigabyte only when processing more than 92,000 proteins. For `Y = {B'}`, the space requirements depend on both the number of proteins and the size of the largest protein. Assuming that the longest protein is still approximately 35,000 amino acids in length, the

proteome would have to contain ~229,000 proteins (or approximately 10x human) before the space requirements exceeded 1 gigabyte.

The estimates of total numbers of loop iterations in the algorithm were based on average parameters of the proteome. This would produce exact estimates if the numbers of iterations of each loop were uncorrelated, but this is not the case in reality. Consider a query protein pair which is in fact a positive pair (known to interact). We would expect that the windows of amino acids on protein A which mediate the interaction are more conserved throughout the proteome and thus have a higher number of similarity hits ($simHitsA_i$) than windows chosen at random. At the same time, proteins which contain that motif are predisposed for interaction with the partner motif and have higher numbers of known partners (neighboursA'). On the other hand, for a negative protein pair, we cannot say much about $simHitsA_i$, but we would not expect both $simHitsA_i$ and neighboursA' to be high at the same time, since that would suggest an interaction in the protein pair, while we are under the assumption that the protein in known to be negative for interaction. Because of this relationship between $simHitsA_i$ and neighboursA', the true number of iterations will be higher than avgSimHits × avgNeighbours for positive pairs (and positive pairs will be more computationally intensive on average), while for negative pairs, the true number of iterations will be lower than avgSimHits × avgNeighbours. This is reflected experimentally by the fact that positive pairs are more time-consuming than negative pairs (section 3.5). This difference between positive and negative pairs suggests that the algorithm modifications might impact the two types of inputs differently, and that we should measure performance improvements on them separately.

3.4 Benchmarking Procedure and Goals

The input data for our performance benchmarks consist of two large sets of protein pairs of 43,591 positive and 100,000 negative pairs in *S. cerevisiae* and 41,678 positive and 100,000 negative pairs in *H. sapiens*. These are the same sets of pairs later used for measuring the classification performance of PIPE. Thus, this workload is precisely the workload we will use when modifying the predictive model and re-validating the classification performance of the new model. Speed improvements on this workload directly relate to our ability to improve PIPE as a classifier.

The PIPE algorithm was executed for each pair of the validation set and for each pair its runtime was taken as the wall time elapsed between the beginning of the computation for the pair and the production of an output pair score. The computation was performed in parallel on a homogenous 28-core computational cluster, distributed between 7 machines, each equipped with 8 GB of RAM and a 4-core Intel Core 2 Quad Q6600 2.4 GHz processor with 8 MB of cache. The operating system-reported memory usage of each PIPE instance was approximately 300 MB, so all data was in main memory and thus RAM capacity should not have a major impact as long as it is greater than approximately 4×300 MB = 1.2 GB per machine. On the other hand, all data would not fit in the CPU cache. We would expect an increase in performance with a larger CPU cache, up to a saturation point where all data is contained in the CPU cache. Some parts of our proposed algorithms improve cache efficiency, so at cache sizes before this saturation point, we expect to observe a speedup. On the other hand, some of the data structure changes result in less CPU operations needed to carry out the same

computations, so we expect to see a benefit from these changes regardless of CPU cache size.

On this parallel cluster, up to 28 pairs were executing simultaneously. The "total runtime" we report for a given experiment is the sum of all individual pair runtimes, not the actual parallel runtime on our cluster. The reported total runtime represents the time that the experiment would take if it were executed serially on one core of a Q6600 processor, and can be used to estimate the parallel runtime on any cluster by dividing it by the number of cores (assuming the cluster has processing cores similar in performance to the Q6600).

Pair runtimes vary widely depending on the source organism and the type of pair (positive or negative). As shown in Table 3 above, the difference by organism can be explained by the different protein lengths (avgLength), average number of similarity hits (avgSimHits), and average number of known interaction partners (avgNeighbours), which themselves vary considerably by organism (see Table 2). The type of pair (positive or negative) affects the runtime for similar reasons, as discussed in section 3.3. Because of these different classes of input data, we are interested in how the modified algorithms affect the performance of each of these types of protein pairs. To this end, we measured runtimes separately each of 4 sets of protein pairs: *S. cerevisiae* positives, *S. cerevisiae* negatives, *H. sapiens* positives and *H. sapiens* negatives.

For each set of protein pairs, we ran both the original and proposed algorithms on all pairs and recorded runtime per pair in each case. Even when the set of protein pairs is restricted to a single organism and a single type of pair, because individual protein lengths and characteristics vary, pair runtimes also vary widely over several orders of

magnitude within one set. To illustrate the overall tendencies of pair runtimes, we plot the cumulative distribution of the runtime for both algorithms on the same axes. Speedups are represented as a general trend of the distribution towards the left of the plot (i.e. towards lower runtimes). The total runtimes for a complete run of the validation set are reported as annotations for each of the cumulative distribution plots. From these, we can calculate a single numerical speedup by taking the ratio of the total runtime of the original algorithm to the total runtime of the proposed algorithm.

From the same set of runtimes measured during the full runs, we also inspect the effect of the proposed algorithm on individual protein pairs. This provides very concrete evidence that, at least for the given pairs, the proposed algorithm does provide a performance benefit. We selected 5 protein pairs at 5 percentile points along the runtime distribution with the original algorithm (10%, 25%, 50%, 90%, 99%). These are meant to demonstrate the effect of the algorithm on pairs of widely-varying difficulty. The last two percentiles explore the performance of the algorithm in the long tail of the distribution (very long-running pairs). We first report the runtimes of the 5 pairs with the original algorithm, and then report the runtime of the same 5 pairs with each proposed modified algorithm.

Care must be taken that our method of selecting pairs of a particular difficulty does indeed achieve that goal. In particular, because the measured runtime is contaminated by measurement noise, it is possible that our selection process did not in fact control the difficulty of the pairs. The 5 pairs in each set could be equally difficult and the variation in observed runtimes could be due to measurement noise only. Then, if we were to re-run the *original* algorithm on the selected "difficult" (long-running) pairs,

we would expect to observe lower runtimes, due to random variation in the measurement noise. This effect is referred to as *regression towards the mean* [49]. If we were to run the *proposed* algorithm on the same 5 pairs, we would still observe lower runtimes, and the difference would be emphasized for the more "difficult" pairs, leading us to an incorrect conclusion about the algorithm. In fact, the experiment is flawed because we assumed that controlling for the original runtime also controls for the difficulty of the problem. Note that our results, when presented for the full sets of proteins (runtime distributions and total runtimes), are not vulnerable to regression towards the mean because they do not attempt to control the difficulty of the pairs – they report aggregate information for all pairs. If measurement noise happens to dominate, we will conclude that the proposed algorithm has no effect, but we will never wrongly conclude that the proposed algorithm is better than it truly is.

The danger of regression towards the mean in practice depends on the magnitude of the measurement noise. The smaller the measurement noise component, the more reliably we can control the difficulty of the test problems by using their runtimes. In our case, measurement noise can stem from insufficient resolution of the timing mechanism (MPI_Wtime, resolution reported by the runtime is 1 μs), or from other processes running on the system which also use the processor. No computational jobs were run on the cluster at the same time as these benchmarking runs. However, there are always a number of background system processes which are usually idle, but which can preempt the processor and slightly inflate the measured runtimes. Empirically, the measurement noise appears to be quite low. We repeatedly ran the same *H. sapiens* pair 17 times and measured its runtime in the same way that runtime is measured for our other experiments.

The runtime had a mean of 17.5 seconds and a variance of 0.2 milliseconds. This is smaller than the smallest runtime of all *H. sapiens* pairs (0.7 ms), while most *H. sapiens* pairs have a runtime in the hundreds of milliseconds (3 orders of magnitude greater than the variance of the measurement).

3.5 Results

We first compare runtimes with the original algorithm between positive and negative pairs. As predicted, in both *S. cerevisiae* (Figure 10 (a)) and *H. sapiens* (Figure 10 (b)), positive pairs have higher runtimes on average than negative pairs. This is due to positive pairs executing on average a higher number of iterations of the inner loops, a result of simultaneously high simHitsA_i and neighboursA' values. Additionally, *H. sapiens* pairs have higher runtimes in general than *S. cerevisiae* pairs, because of the much higher value of avgSimHits. The *H. sapiens* pairs took the longest to execute in total (38.8 h and 75.5 h for positive and negative *H. sapiens* pairs, respectively, compared to 16.1 h and 17.0 h for *S. cerevisiae* pairs). Note that due to the long tail of the distribution, we have only plotted the distribution between 0 and 1 second. Depending on the set, some proportion of pairs took longer than 1 second to complete, and this can be calculated from the last data point at 1 second. For example, approximately 40% of *H. sapiens* positives have a runtime greater than 1 second. In subsequent plots of runtimes with the proposed algorithm, due to the speedup, a much smaller proportion of pairs (typically < 10%) are outside the range of the x-axis.

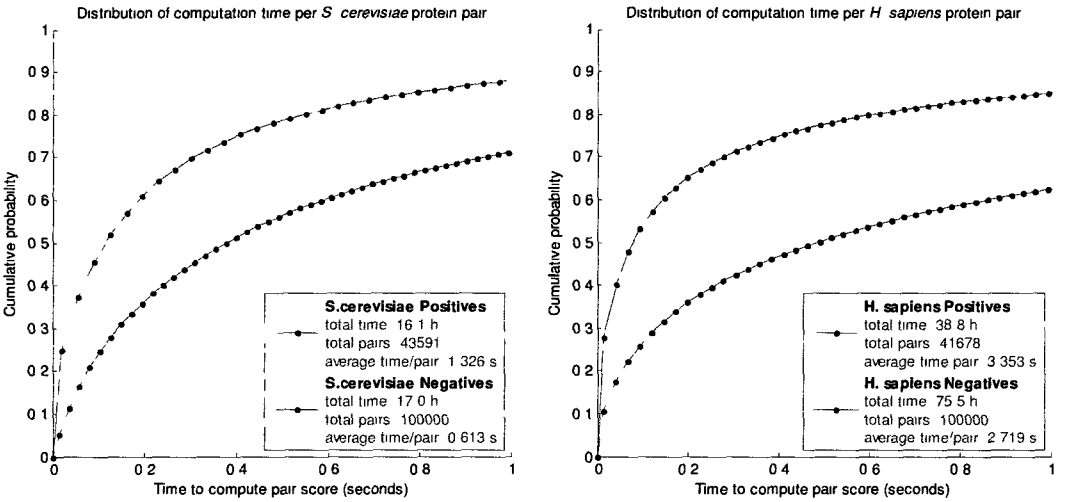


Figure 10: Distributions of pair runtimes for the original algorithm. (a) 43,591 positive and 100,000 negative *S. cerevisiae* protein pairs and (b) 41,678 positive and 100,000 negative *H. sapiens* protein pairs

We now focus on 5 positive and 5 negative protein pairs in each of *S. cerevisiae* and *H. sapiens*. These pairs were chosen to have runtimes in various percentiles of the original runtime distributions (10, 25, 50, 90, 99), so that they are representative of the overall set of runtimes. The 50^{th} percentile runtimes are the median runtimes for each set. Because the runtime distributions are long-tailed, the last two pairs were chosen in the very high percentiles (90^{th} and 99^{th}) so that we can analyze the behaviour of the algorithm changes in this region of very slow pairs. This allows us to ensure that the performance gains are observed for the entire range of protein pairs. Furthermore, the PIPE authors have reported that long-running pairs have impaired the ability of a computational cluster to load-balance PIPE jobs, extending the total runtime. Easier pairs would complete very quickly, leaving only one processor to execute a very difficult pair. The narrower the range of runtimes, the less time is likely to be spent at the end of a large batch of pairs waiting for a longer-running pair to finish. Thus, speedups in the long tail of runtimes

have an added practical benefit of facilitating load-balancing of PIPE jobs on computational clusters and reducing overall runtime.

Table 5 lists the chosen protein pairs in each set, and their runtimes with the original algorithm. These pairs represent a wide range of runtimes, spanning several orders of magnitude from 9 ms for a very fast *H. sapiens* negative (P06865 Q13323) to 38 seconds for a very slow *H. sapiens* negative (Q92793 Q96JA1). The trends observed in the complete runtime distributions are still present: for the same percentiles, positive pairs tend to execute more slowly than negative pairs, with *H. sapiens* pairs tending to have longer runtimes than *S. cerevisiae* pairs.

Type of protein pair	**Runtime percentile**	**Protein A**	**Protein B**	**Runtime (original algorithm)**
S. cerevisiae Positive	10%	YHR085W	YMR242C	41 ms
	25%	YJL203W	YNR053C	118 ms
	50%	YAR007C	YPR175W	392 ms
	90%	YCR077C	YPL240C	3259 ms
	99%	YDL153C	YHR135C	13152 ms
S. cerevisiae Negative	10%	YIL012W	YMR183C	12 ms
	25%	YDR380W	YGL260W	37 ms
	50%	YJR155W	YNL073W	130 ms
	90%	YIL146C	YIL172C	1229 ms
	99%	YGL227W	YOR178C	7220 ms
H. sapiens Positive	10%	P52565	Q11206	26 ms
	25%	O00590	Q99616	101 ms
	50%	P27797	P61769	495 ms
	90%	O00303	P23443	7102 ms
	99%	P12931	P48023	35882 ms
H. sapiens Negative	10%	P06865	Q13323	9 ms
	25%	A6NFN9	P02771	24 ms
	50%	Q309B1	Q6QNK2	90 ms
	90%	A0MZ66	P18803	1883 ms
	99%	Q92793	Q96JA1	38071 ms

Table 5: Runtimes for 5 representative pairs from each test set. The test sets considered were *S. cerevisiae* positives, *S. cerevisiae* negatives, *H. sapiens* positives and *H. sapiens* negatives.

We now turn our attention to the first proposed changes to the algorithm, the changes in input data representation of the similarity lists and known interaction lists, described in detail in section 3.2. We expect that this first set of changes will impact all pairs almost equally, because the optimizations are based on more efficient memory access patterns which would apply to the computations for any types of pairs.

At the beginning of this section, we compared positive and negative pair runtimes in the same axes, with a separate plot for each of *S. cerevisiae* and *H. sapiens*. Here, to evaluate the effect of the modified algorithm, we show the *original and proposed algorithm* distributions in the same plot, for a given set of protein pairs. We show a separate plot for each of *S. cerevisiae* positive pairs (Figure 11 (a)), *S. cerevisiae* negative pairs (Figure 11 (b)), *H. sapiens* positive pairs (Figure 12 (a)) and *H. sapiens* negative pairs (Figure 12 (b)). As in the original distribution plots, the red traces represent positive pairs, while the blue traces represent negative pairs. Note that the modified algorithm shifts the distribution above and to the left, indicating a trend toward lower pair runtimes. The speedup on total runtime is 2.9, 3.3, 3.6 and 8.3, for each of the protein pair sets, respectively, relative to the total runtimes with the original algorithm. With the exception of *H. sapiens* negative pairs (the last set of pairs), the speedup is approximately the same. This is expected since the algorithm modifications only improve time complexity by a constant factor, which is independent of the characteristics of the input data.

The variability of speedup across test sets, especially the high speedup on total runtime of *H. sapiens* negative pairs, may be explained by the very long tails of the original runtime distributions. When we removed from each set the 1% of pairs with the longest original runtime, the speedup on total runtime for the remaining pairs was 2.4,

2.3, 2.5 and 2.6 for each set, respectively (compare to 2.9, 3.3, 3.6 and 8.3 for the full sets), which is in better agreement with the constant speedup hypothesis. The distribution of original runtimes for *H. sapiens* negative pairs is particularly long-tailed (as shown by the 99th percentile value of 38,071 ms in Table 5, the largest of all test sets). However, this means that our analysis does not properly account for the effect of the modifications on pairs in the top 1% of original runtimes, and suggests that the effect of the changes could be characterized in more detail. A future study could examine the effect of control variables other than original runtime, such as protein length or other characteristics of the protein pairs, which may be correlated with speedup. Since the negative set is generated by randomly sampling the set of all pairs, it is possible that some of the negative pairs are in fact previously unreported positive pairs, explaining their longer runtime and their deviation from the behaviour observed in the lower runtime percentiles of mostly negative pairs. The predicted interaction score of each protein pair may therefore also be a useful control variable for predicting speedup.

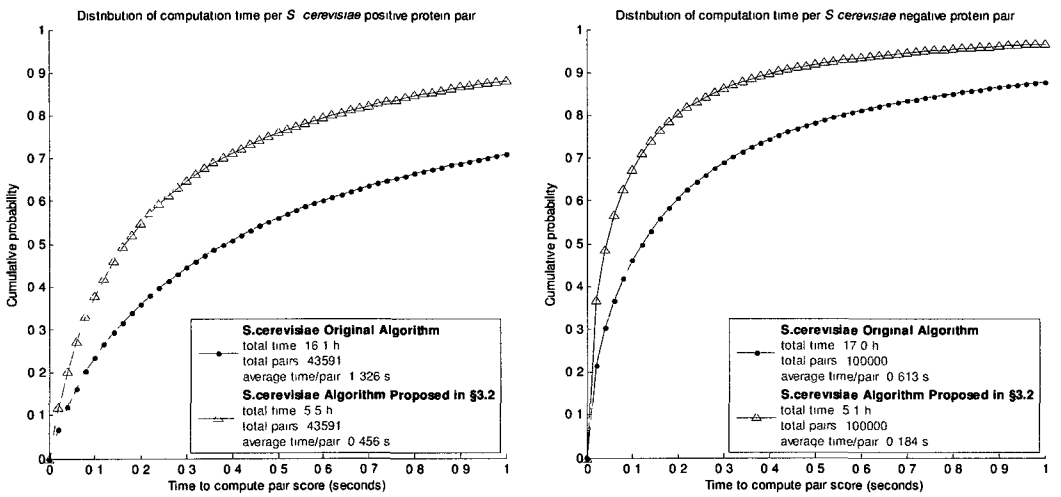


Figure 11: Effect of algorithm changes proposed in section 3.2 on runtime distribution of positive and negative *S. cerevisiae* protein pairs

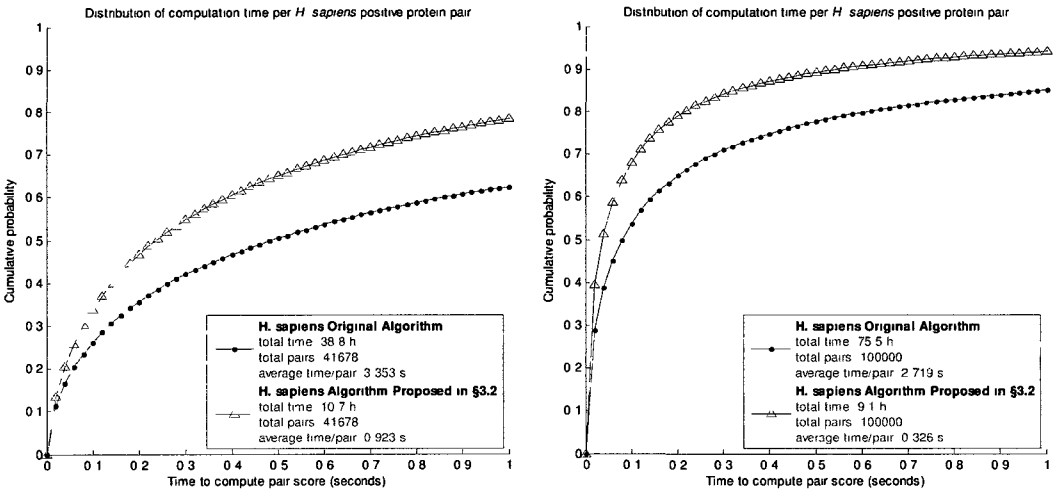


Figure 12: Effect of algorithm changes proposed in section 3.2 on runtime distribution of positive and negative *H. sapiens* protein pairs

These performance improvements are reflected in the individual runtimes of the 5 representative pairs for each set of pairs (Table 6). The names of the protein pairs were omitted for brevity, but they are the same protein pairs as those shown in Table 5. In general, higher percentile pairs benefited more from the improvements. One possible explanation is that these pairs spend more time in the inner loops, spending more processing time on the same small set of data. This means that they have better temporal locality of reference, and the CPU cache is able to better amortize the startup cost of fetching the data from main memory. By contrast, the slow pairs iterate over a small amount of data spread across main memory and do not benefit from the CPU cache as much, or from improvements to the cache efficiency of the algorithm.

However, there are still anomalies in the *H. sapiens* negatives test set. First, the runtime of the *H. sapiens* negatives 10^{th} percentile pair increased from 9 ms to 10 ms. This may be a floor effect, in that both algorithms performed similarly, and the difference is a result of measurement noise alone [49]. Even if the algorithm were solely responsible

for the increase in runtime, a small increase for these short-running pairs would not greatly affect the total runtime for the test set. Second, the speedup of the 99th percentile pair does not follow the pattern of increasing speedups for larger original runtimes. Since this anomaly is limited to the highest percentile, this again points to additional factors impacting speedup for protein pairs in the long tail of original runtime (as discussed above)..

Type of protein pair	Runtime Percentile	Runtime (original algorithm)	Runtime (changes proposed in §3.2)	Speedup
S. cerevisiae Positive	10%	41 ms	24.1 ms	1.7
	25%	118 ms	71.3 ms	1.7
	50%	392 ms	248.8 ms	1.6
	90%	3259 ms	876.2 ms	3.7
	99%	13152 ms	2946.8 ms	4.5
S. cerevisiae Negative	10%	12 ms	4.6 ms	2.7
	25%	37 ms	18.2 ms	2.0
	50%	130 ms	52.5 ms	2.5
	90%	1229 ms	332.4 ms	3.7
	99%	7220 ms	2127.2 ms	3.4
H. sapiens Positive	10%	26 ms	24.8 ms	1.0
	25%	101 ms	51.7 ms	2.0
	50%	495 ms	145.0 ms	3.4
	90%	7102 ms	1526.4 ms	4.7
	99%	35882 ms	7985.5 ms	4.5
H. sapiens Negative	10%	9 ms	10.0 ms	0.9
	25%	24 ms	17.0 ms	1.4
	50%	90 ms	42.8 ms	2.1
	90%	1883 ms	369.3 ms	5.1
	99%	38071 ms	15327.2 ms	2.5

Table 6: Effect of algorithm changes proposed in section 3.2 on runtimes of the 5 representative. The protein pairs used for each measurement are given explicitly in Table 5.

By combining the input data representation changes proposed in section 3.2 and the space-time trade-off proposed in section 3.3, we can achieve even better speedups. In *S. cerevisiae*, using the combined changes, we observed an average speedup of 8.1 for positive pairs (Figure 13 (a)) and 8.9 for negative pairs (Figure 13 (b)). In *H. sapiens*, the

average speedup was 4.0 for positive pairs (Figure 14 (a)) and 14.5 for negative pairs (Figure 14 (b)). In both cases, the speedup is relative to the total runtime with the original algorithm. In almost all cases, more than 99% of the pairs in each set completed in less than 1 second. This thinner tail of the distribution further reduces the time spent waiting for a very long pair to finish at the end of a large batch of protein pairs, facilitating load-balancing of PIPE jobs on parallel clusters.

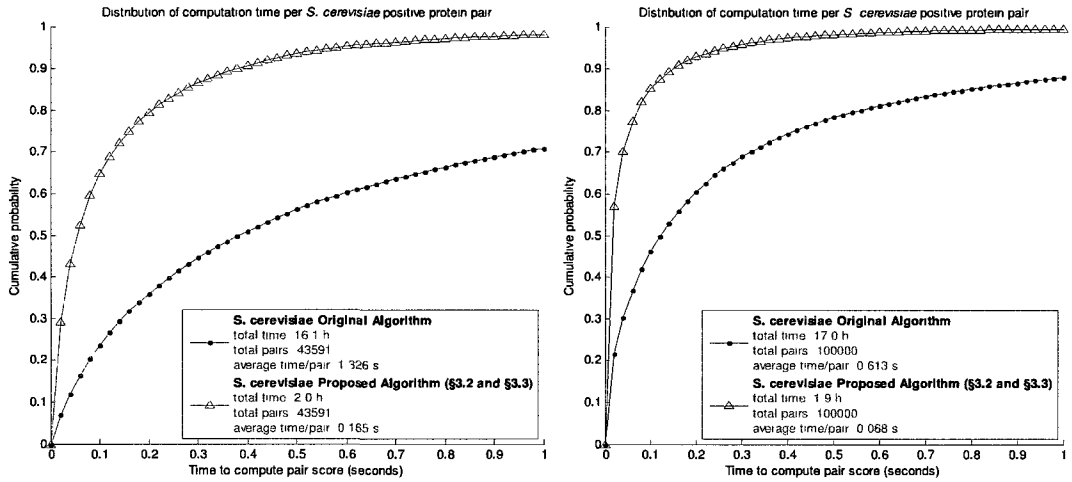


Figure 13: Effect of all proposed algorithm changes (sections 3.2 and 3.3) on runtime distribution of positive and negative *S. cerevisiae* protein pairs

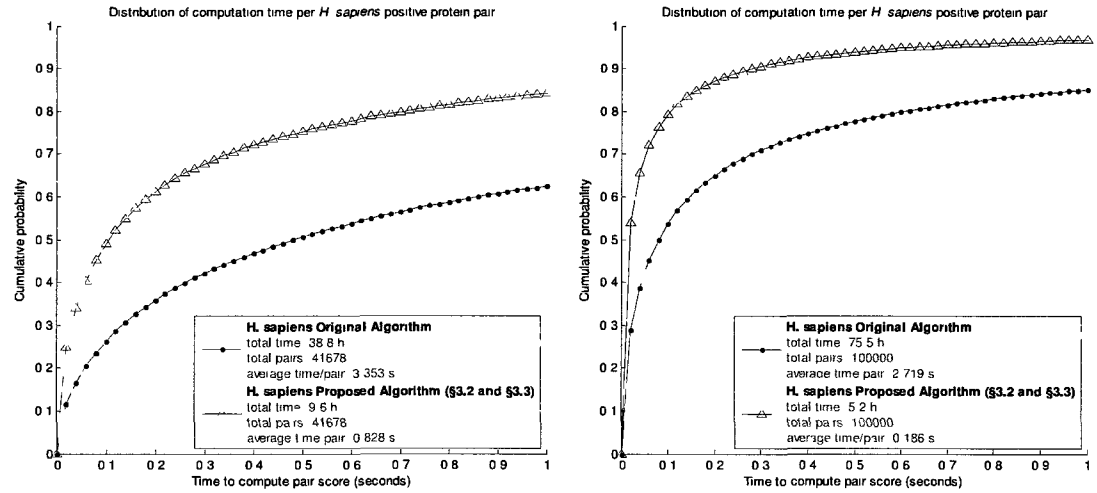


Figure 14: Effect of all proposed algorithm changes (sections 3.2 and 3.3) on runtime distribution of positive and negative *H. sapiens* protein pairs

These speedups can be seen for the 5 representative pairs in each of the 4 sets of protein pairs (Table 7). Compared to the speedups achieved using only the changes in section 3.2 (Table 6), here the data-dependent nature of the optimization plays a role in the observed speedups. For *S. cerevisiae*, positive pairs, which have higher simHits values, have larger speedups (in the range 3.8 – 32.9), while negative pairs benefit less from the optimization (speedups in the range 2.1 – 11.5). *H. sapiens* positive pairs also follow this pattern, but the behaviour on *H. sapiens* negative pairs is more difficult to explain. When we removed the top 1% of pairs in each set by original runtime, the total runtime speedups were 6.6, 6.3, 2.8 and 4.6, respectively (compare to 8.0, 8.9, 4.0 and 14.5 for the full sets). This suggests that in *H. sapiens*, very long-running pairs contribute the most to total runtime speedup, while in *S. cerevisiae* the speedup is more even across all percentiles. Again, original runtime proves insufficient as the only control variable for *H. sapiens* pairs, particularly in the upper percentiles.

Type of protein pair	Runtime Percentile	Runtime (original algorithm)	Runtime (all changes, §3.2 and §3.3)	Speedup
S. cerevisiae Positive	10%	41 ms	10.8 ms	3.8
	25%	118 ms	33.2 ms	3.6
	50%	392 ms	100.4 ms	3.9
	90%	3259 ms	249.9 ms	13.0
	99%	13152 ms	399.6 ms	32.9
S. cerevisiae Negative	10%	12 ms	5.8 ms	2.1
	25%	37 ms	6.6 ms	5.6
	50%	130 ms	25.9 ms	5.0
	90%	1229 ms	106.9 ms	11.5
	99%	7220 ms	821.0 ms	8.8
H. sapiens Positive	10%	26 ms	13.9 ms	1.8
	25%	101 ms	15.4 ms	6.6
	50%	495 ms	42.0 ms	11.8
	90%	7102 ms	254.8 ms	27.9
	99%	35882 ms	447.7 ms	80.1
H. sapiens Negative	10%	9 ms	6.3 ms	1.5
	25%	24 ms	13.8 ms	1.8
	50%	90 ms	46.9 ms	1.9
	90%	1883 ms	38.5 ms	49.0
	99%	38071 ms	8893.4 ms	4.3

Table 7: Effect of all proposed algorithm changes (sections 3.2 and 3.3) on runtimes of the 5 representative pairs.

Nevertheless, these results demonstrate that the proposed algorithm changes reduce total runtime of test sets in both *S. cerevisiae* and *H. sapiens*, for positive and negative pairs. In *S. cerevisiae*, the speedup is consistent for pairs of varying difficulty (original runtime), while in *H. sapiens* it appears that very long-running pairs are responsible for the bulk of the speedup on total runtime. The total single-core runtimes for 143,591 validation *S. cerevisiae* pairs was reduced from 33.1 hours to 3.9 hours, while the time to run the 141,678 validation *H. sapiens* pairs was reduced from 114.3 hours to 14.8 hours. On our 28-core cluster, these runs now each complete in less than an hour. This greatly facilitates modifications to the classification model of PIPE because

validation runs, which provide crucial feedback to the experimenter, can be performed much more quickly.

The total runtime was reduced the most for negative pairs. This is particularly useful for all-to-all interaction screens, where we expect the vast majority of pairs tested to be negative pairs. In particular, if we assume that the runtime distributions of positive and negative pairs are representative of the true distributions for all positive and negative pairs, we can use the average per-pair times to estimate the total expected time before and after the proposed modifications. For example, in *H. sapiens*, the number of positive pairs is taken as Stumpf's estimate (650,000) and the total number of negative pairs is calculated as the total number of pairs ($\frac{22{,}513^2+22{,}513}{2}$) minus the 650,000 positive pairs, which is approximately 253 million pairs. Table 8 shows that an all-to-all screen would complete 14.5 times faster with the proposed changes than with the original algorithm. Not surprisingly, the overall speedup is almost equal to the speedup on negative pairs, because the set of all possible pairs is, in fact, dominated by negative pairs. While the 550 days required with a single core means all-to-all screens remain out of reach for the bioinformatician with only their office computer at their disposal, on a 28-core (7-machine) cluster this time was reduced to 20 days. It is likely that, at least for the exhaustive all-to-all screen in *H. sapiens*, the *in silico* experiment using PIPE is much less costly than a wet lab experiment, in terms of both time and physical resources.

Type of pairs	Number of pairs	Average runtime per pair	Total time
Original Algorithm			
Positives	650,000	3.353 s	605 h
Negatives	253,000,000	2.719 s	190,874 h
		Total time (single core):	7,978 days
		On a 28-core cluster:	284 days
Proposed Algorithm			
Positives	650,000	0.828 s	150 h
Negatives	253,000,000	0.186 s	13,072 h
		Total time (single core):	550 days
		On a 28-core cluster:	20 days

Table 8: Estimated times for an all-to-all interaction screen in *H. sapiens* for the original and proposed algorithms

4 Improved Classification Performance of PIPE using Sequence Annotations

4.1 Introduction

In Park's review of sequence-based protein-protein interaction predictors [15], PIPE was shown to have the best classification performance out of all four tested classifiers, on both *S. cerevisiae* and *H. sapiens* test sets. However, Park's consensus method which combined all four classifiers outperformed PIPE, suggesting that there are still improvements to be made in PIPE's prediction model.

There are many potential sources of errors in PIPE predictions. As discussed in sections 1.1 and 1.2, we will mainly focus on sources of false positive errors since reducing the false positive rate (FPR) directly increases specificity, which is crucial for organism-wide protein-protein interaction prediction. Errors can be reduced either by introducing additional model features (additional input data) or by refining the model's use of existing data (protein sequences). When modifying the model to require additional data, it will be important to consider the availability of this data across the organisms of interest (*S. cerevisiae* and *H. sapiens*). Availability of data becomes a greater issue when applying PIPE to less-studied organisms with scarce proteomic data (such as *M. musculus*), which are precisely the organisms for which PIPE can provide the most interesting results.

4.1.1 Protein Functional Annotations

Protein sequences exhibit re-occurring patterns which have specific functional roles. Sometimes, the role of these regions is to enable interaction of the protein with

other proteins. For example, certain protein domains (such as SH3 [50]) are known to mediate protein-protein interactions (PPIs). More often, sequence regions have specific functional roles, but are not associated with PPIs, or their properties suggest the absence of PPIs within that region of the protein. Two examples of such well characterized sequence motifs are signal peptides and transmembrane regions. Signal peptides are removed from the protein upon maturation, so they cannot be involved in the PPIs of the mature protein. Transmembrane regions are bound in cell membranes and are physically isolated from other proteins, making it unlikely that they mediate PPIs. Thus, because the presence of these features in a particular protein region may bias the likelihood of a PPI in that region, annotations of these two types of protein regions could be useful as an additional sequence feature to improve PIPE predictions. Signal peptides and transmembrane regions serve as a pilot study of the applicability of sequence annotations for improving PIPE predictions – other types of sequence regions, such as DNA-binding motifs, may also be informative with respect to PPIs.

Signal peptides are short linear regions of amino acids (20-25 amino acids) located at the N-terminal of the protein (beginning of the sequence) which control the physical targeting of proteins within the cell or to the exterior of the cell [51]. There are several protein translocation pathways, controlling export across the plasma membrane, endoplasmic reticulum membrane, import into the mitochondria and other cellular compartments. After directing the protein to the appropriate pathway, signal peptides are cleaved from the beginning of the amino acid chain by pathway-specific mechanisms. Because signal peptides are no longer present in the mature protein, they most likely do

not participate in the functional interactions of the protein once it has been translocated, and are unlikely to mediate protein-protein interactions.

Transmembrane regions are also short linear regions of amino acids, which are found in proteins which become embedded in organelle or cell membranes upon maturation (integral membrane proteins). These linear sequence regions are composed of hydrophobic amino acids, which favour interaction with the hydrophobic membrane lipid bilayer [52]. Some examples of integral membrane proteins are transporters and ion channels. In single-pass transmembrane proteins, the protein passes through the membrane only once and is composed of an intracellular region, the transmembrane region, and an extracellular region. Multi-pass transmembrane proteins penetrate the membrane at multiple positions along the amino acid chain and have several transmembrane regions, bounded by regions which alternate between the intracellular and extracellular space. In both cases, the transmembrane regions are tightly packed either against membrane phospholipids or against other transmembrane regions of the same protein. Because of this, these regions are physically isolated from any proteins inside or outside the cell, and are unlikely to be the mediators of the membrane proteins' interactions.

PIPE's scoring of a particular protein region for interaction is partially based the re-occurrence of its amino acid sequence in the proteome of the organism (see section 2.3.1). Because these sequence features re-occur in the proteome (for example, multiple proteins may bear the same signal peptide because they are tagged for translocation to the same organelle), PIPE finds a large number of similarity hits for these regions, increasing the number of hits in the PIPE matrix (or "landscape") for these non-interacting regions.

This increases the overall PIPE score, suggesting an interaction, but is in fact contributing to false positives because these protein regions may not in fact mediate interaction.

Experimental assays can detect these protein regions and published results are curated and collected in large protein databases such as Uniprot [28]. According to current experimental data in Uniprot, these two types of regions are present across both prokaryotes and eukaryotes, and are relatively abundant across proteomes. Approximately 5% of *S. cerevisiae* proteins and 17% of *H. sapiens* proteins have experimentally-confirmed signal peptides, while 20% of *S. cerevisiae* proteins and 23% of *H. sapiens* proteins have experimentally-confirmed transmembrane regions.

4.1.2 Prediction of Functional Annotations

For less-studied organisms (other than *S. cerevisiae* and *H. sapiens*), sequence annotations for signal peptides and transmembrane regions may be scarce. There are specific computational prediction tools (per type of sequence features) which can be applied to proteins for which experimental data is not yet available. However, the results of these prediction tools must be interpreted with care because the underlying models are themselves trained from the scarce proteomic data. Errors in the experimental data, or shortcomings of the prediction model, lead to errors in the predicted sequence annotations, which can adversely affect the performance of PIPE if it integrates these erroneous predictions.

For signal peptides, SignalP [53] is commonly cited as a strong prediction tool for signal peptides [54]. This tool comes in two variants: SignalP-NN, which is based on a neural network applied to a sliding window of amino acids, and SignalP-HMM, which fits a hidden Markov model to the full sequences. Emanuelsson et al. [54] recommend

using SignalP-NN, and show that this variant has slightly higher sensitivity (99.4%) compared to SignalP-HMM (98.4%) for the same false positive rate (~3%). We applied SignalP-NN to our database of 6,716 *S. cerevisiae* proteins and found 722 proteins with predicted signal peptides (compare to 310 proteins with signal peptide annotations in Uniprot). In *H. sapiens*, SignalP-NN reported signal peptides in 5,716 out of 22,513 proteins (compared to 3,427 in Uniprot).

A group from the same university has also published a tool for transmembrane region prediction, called TMHMM [55]. This tool is based exclusively on a hidden Markov model whose architecture mimics the topology of transmembrane proteins. In particular, this method has been previously applied to whole-genome data sets in a large set of organisms [56], including *S. cerevisiae*. We applied the tool on our own set of 6,716 *S. cerevisiae* proteins and obtained transmembrane annotations for 1,477 proteins (compared to 1,407 in Uniprot). In *H. sapiens*, TMHMM reported transmembrane regions in 6,258 out of 22,513 proteins (5,484 in Uniprot).

4.1.3 Validation of Functional Annotations Using Binding Site Data

We hypothesize that signal peptides and transmembrane regions can be used to improve PIPE predictions because these protein regions are less likely to be involved in PPIs. Specifically, we hypothesize that signal peptides and transmembrane regions do not overlap with those regions of proteins which mediate PPIs (binding sites). Before working this assumption into the PIPE model, we wish to validate our hypothesis based on experimentally-determined binding sites. We use two sources of binding site data for this purpose, and validate our assumptions on signal peptides and transmembrane regions against both of these databases of binding sites.

DOMINO [57] is a database of protein-peptide interactions which includes the specific linear regions of each protein which are involved in the interaction. The data is collected from literature and represents a wide variety of experimental techniques, but a large proportion of the data originates from deletion experiments. In some cases where the experimental method finds interactions partners of a peptide, the reported results were expanded to include all proteins containing that peptide, and the binding site was taken to be that peptide's sequence position within the protein. This database was previously used to validate the use of PIPE as a binding site prediction tool [58]. The latest version of the DOMINO database (2009/10/22) contains a total of 14,539 binary interaction records between 3,048 proteins across 79 organisms. In some cases, DOMINO references proteins which are unreviewed (unconfirmed) in Uniprot, contains data for only one side of the interaction (i.e. only one of the proteins in the interaction are annotated with binding region information), or references a particular protein domain without specifying the exact amino acid indices of the interacting region. We filtered out these incomplete records, and merged binding site information by protein, which resulted in binding sites for 225 *S. cerevisiae* and 649 *H. sapiens* proteins.

PiSite [59] uses 3D structures of protein complexes from the PDB protein structure database [10] to determine which amino acids in a protein are seen in physical proximity to amino acids from interaction partners. This provides an interaction index along a protein's sequence which indicates whether a given amino acid has ever been observed to be physically involved in a protein-protein interaction. The version of PDB used by PiSite (July 2008) contains 49,175 structures, but only 25,220 are structures of protein complexes. In addition, these represent many different organisms, and often the

same protein is present in multiple complexes. For each protein, we combined the interacting amino acids from all complexes in which it is present, and constructed a set of binding sites for 370 *S. cerevisiae* and 1,648 *H. sapiens* proteins.

4.1.4 Sequence Window Uniqueness

The output value for a particular cell of the PIPE matrix is calculated from the two corresponding sequence windows of the query proteins, one from each protein (refer to 2.3.1). The two sequence windows are compared to the organism's proteome, which results in two lists of proteins which contain similar sequence windows, one for each query protein. PIPE then searches every pair of proteins, one from each list, against the database of known interacting pairs. Each protein pair which is known to interact contributes evidence toward the hypothesis that the original two query windows participate in a PPI. PIPE increases the value of the output cell by 1 for each known pair. Higher output values more strongly suggest an interaction between the original two sequence windows, while low values suggest no interaction.

Sometimes, proteins contain highly re-occurring sequence windows which do not participate in PPIs. Signal peptides (section 4.1.1) are one example of re-occurring sequence motifs which do not mediate interaction. If both query proteins have signal peptides, they will also have sequence similarity to all other proteins which have that same signal peptide. Many of these proteins will participate in interactions which are not, in fact, mediated by the signal peptides. This leads to an inflated number of PIPE hits for the areas of the PIPE matrix corresponding to the signal peptides, while it is very unlikely that the signal peptide regions themselves mediate an interaction between the query proteins.

Another type of sequence motif which is susceptible to the same effect is "low-complexity regions". These regions are usually long repetitions of the same amino acid (aspartic acid, "D" or asparaginine, "N"), pairs of similar amino acids ("DE"), or more generally regions where the composition of amino acids is heavily biased towards certain amino acids. Because these sequence regions are composed of a more restricted set of amino acids, they are highly self-similar, particularly when they are composed of a long string of the same amino acid. This leads to increased numbers of false PIPE hits in those regions, much like the re-occurring signal peptides lead to false hits in those regions of the PIPE matrix.

One example of this is the *S. cerevisiae* protein pair YBR030W-YGL203C (Figure 15), where the sequence of YBR030W between positions 418 and 427 is biased towards glutamic acid (E) and the sequence of YGL203C between positions 481 and 600 is biased towards aspartic and glutamic acid (D and E). This pair of low-complexity regions, often found throughout the proteome, causes a broad area of high PIPE hits in the matrix (400-1000 hits), but which does not truly mediate interaction. YBR030W is "Ribosomal N-lysine methyltransferase 3," involved in ribosome assembly and localized to the nucleus, while YGL203C is "Carboxypeptidase KEX1," a membrane-bound protein, involved in C-terminal protein processing in the trans-Golgi network (annotations from Uniprot entries for each protein) with active sites at amino acids 198, 405 and 470. The protein regions with heightened PIPE hits are unlikely to interact (for YGL203C, the indicated region does not contain any of the active sites), and in fact the two proteins overall are unlikely to interact, because they participate in different cellular functions and are localized in different parts of the cell.

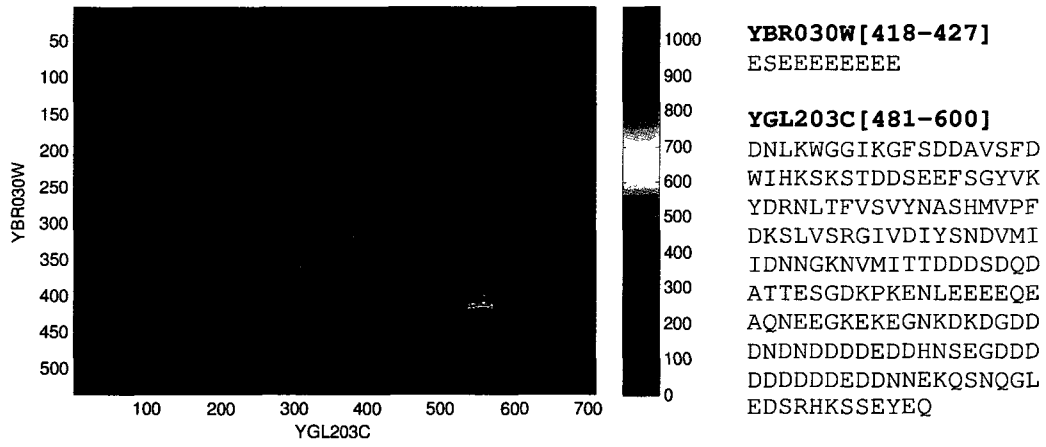


Figure 15: PIPE matrix for the *S. cerevisiae* protein pair YBR030W-YGL203C. Regions of low-complexity sequences (YBR030W[418-427] and YGL203C[481-600]) on each protein cause large numbers of spurious hits in the PIPE matrix.

4.2 Methods

4.2.1 Functional Annotation Hypothesis Testing

We hypothesize that the presence of a particular type of annotation in a protein sequence region indicates that the region is unlikely to mediate interaction. To test this, we consider the events "amino acid is in a PPI binding site" and "amino acid is annotated with a particular sequence feature" for each individual amino acid of all proteins in our sequence database. Each event corresponds to a binary random value which indicates whether the given amino acid is or is not in a binding site, and is or is not annotated with a particular sequence feature, respectively. In this context, our hypothesis is that the binding site and sequence feature events are negatively correlated. If that is truly the case, one event can be used to predict the absence of the other. In our case, we are interested in using the sequence feature event to predict the absence of the binding site event. Our study is repeated for both types of sequence annotations of interest: signal peptides and transmembrane regions.

In practice, we count the numbers of amino acids which are involved in binding sites, which are annotated with a particular sequence feature, and which satisfy both conditions simultaneously. If the number of amino acid which are both in binding sites and annotated with a particular sequence feature is lower than the number of such amino acids expected by chance, then we can conclude that there is a negative correlation between the two events. For example, if 1% of amino acids are observed to be in binding sites, and 0.5% of amino acids are observed to be in transmembrane regions, we would expect $1\% \times 0.5\% = 0.005\%$ of amino acids to participate in both, under the assumption that the two events are independent. However, if the empirical percentage is lower than 0.005%, the data suggest that these events are negatively correlated. Clearly, the significance of this result depends on the size of the data set, and this will be taken into account in our experiments as part of the statistical hypothesis test.

Furthermore, if we assume that the binding site events for each individual amino acid are pairwise independent and the sequence feature events for each individual amino acid are also pairwise independent, then the counting experiment is a series of Bernoulli trials and the amino acids counts are random variables which follow a binomial distribution. This allows us to perform a statistical hypothesis test and obtain a P-value for our conclusions [60]. Because the χ^2 test for independence is not suitable when some event frequencies are very low (less than 5 counts), we perform the more computationally-intensive Fisher's exact test for independence between two categorical variables. We use a left-tailed test where our null hypothesis is that the events are independent and our alternative hypothesis is that they are negatively correlated, and report the P-value of rejection of the null hypothesis. This is the probability of the events

being independent despite the observed amino acid counts. A lower value indicates a stronger assertion of the (negative) correlation between the two variables.

4.2.2 Improved Classification using Functional Annotations

Assuming our hypotheses are confirmed, we wish to reduce the number of PIPE hits originating from sequence regions which are unlikely to mediate PPIs. For each protein in the PIPE sequence database, we look up its entry in Uniprot and replace every amino acid which is annotated to be in a signal peptide or transmembrane region with the virtual "X" amino acid (Figure 16). In the PAM120 matrix, which is used by PIPE for sequence comparisons, "X" signifies a sequence position with an unknown amino acid. The PAM120 matrix row and column for "X" has very low similarity scores, resulting in a low overall PAM score for sequence windows containing "X" amino acids. Only a pre-processing step of the sequence database is necessary for this modification of the model. No modifications to the PIPE algorithm are necessary, but the sequence comparison precomputation must be re-executed after constructing the modified sequence database. This pre-computation needs to be performed only once, after which any number of protein pairs may be tested using PIPE.

Protein: YAR071W

Original sequence: MLKSAVYSILAASLVNAGTIPLGKLSDIDKIGTQ...

Excerpt from Uniprot entry:

```
FT   SIGNAL        1      17       By similarity.
```

Sequence annotation: MLKSAVYSILAASLVNAGTIPLGKLSDIDKIGTQ...

■ Signal peptide

Masked sequence: XXXXXXXXXXXXXXXXXGTIPLGKLSDIDKIGTQ...

Figure 16: Example use of Uniprot annotations for masking protein sequences prior to PIPE prediction

4.2.3 Improved Classification using Sequence Window Uniqueness

Highly re-occurring sequence regions which do not mediate PPIs lead to false hits in the PIPE output matrix (see section 4.1.4). We wish to reduce the impact of these regions, by reducing the weight given to PIPE hits within regions which are highly re-occurring in the proteome. This is a generalization of the method presented in section 4.2.2, where we excluded specific re-occurring sequence regions (i.e. signal peptides and transmembrane regions) from contributing to the PPI hits in the corresponding areas of the PIPE matrix. Now, we use PIPE itself to measure re-occurrence of sequence windows and to weigh them accordingly.

Recall that two factors contribute to the final number of PIPE hits in a particular cell of the output matrix: the degree of re-occurrence of the two corresponding sequence windows, and the number of hits in the known interactions database involving proteins which are sequence-similar to the query proteins. The higher the degree of re-occurrence of the query sequence windows, the longer the candidate list of possibly known interacting pairs, and the higher the likelihood of random hits in the known interactions database. This can lead to false positive predictions. We therefore propose to compensate

for this effect using the length of the similarity lists for each sequence window. Longer similarity lists will require more hits in the database of known interactions to provide the same level of evidence for the interaction. Conversely, two relatively unique windows, with short similarity lists, will probe the known interactions database a small number of times, and the few resulting hits will be considered strong evidence for the interaction between the query windows.

Specifically, when calculating the PIPE matrix, instead of assigning the total number of hits in the known interacting pairs database to each pair of input windows (A_i, B_j), we store the proportion of hits which were found in the database, relative to the total possible number of pairs between proteins similar to A and B. Figure 17 shows an example of this for a pair of windows in the case where (a) the windows are relatively unique (few similarity matches), and (b) the windows re-occur often throughout the proteome. While each case results in 6 hits in the known interacting pairs database, the first case represents strong evidence that A_i and B_j interact, because there are experimentally-validated interactions between almost all of the pairs of proteins which contain windows similar to A_i and B_j. On the other hand, this proportion of pairs in the second case is much lower, suggesting that A_i and B_j do not interact (e.g. perhaps the known interaction A′–B′ was caused by other regions in those proteins). Because both cases have the same number of known interacting pairs between the two similarity lists, PIPE would originally assign the same value (6) for the matrix cell corresponding to (i, j). To account for the sizes of each similarity list, we assign the ratio $\frac{NumberOfKnownPairs}{TotalPossibleKnownPairs} = \frac{NumberOfKnownPairs}{SizeOfSimilarityListA \times SizeOfSimilarityListB}$ to matrix cell (i, j), which is numerically greater for more unique sequence windows. In the

example from Figure 15, we assign 0.50 to the cell in the first case and 0.14 in the second case, representing the much stronger evidence for interaction of the first case. When either similarity list is empty, this technically represents a lack of evidence in either direction (the denominator above is 0) – that is, we do not know anything at all about that particular sequence window. In that case, we default to assigning a value of "0." This represents the low probability of two sequence windows chosen at random to interact, because the PAM120 score cut-off was chosen such that two random sequence windows would be highly unlikely to be considered as "similar" and appear in each others' similarity lists.

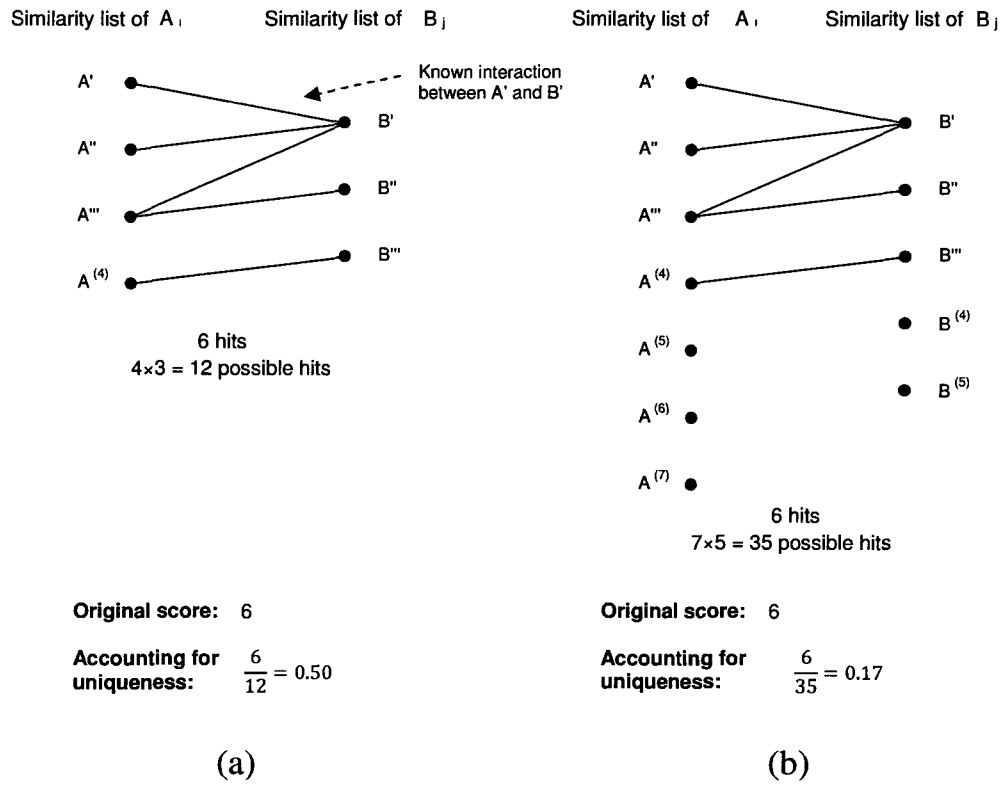


Figure 17: Accounting for sequence window uniqueness when finding evidence of interaction between two sequence windows. The calculation of the strength of the evidence is shown for two windows A_i and B_j which are (a) relatively unique and (b) widely re-occurring

4.2.4 Measurement of Classification Performance

In order to measure changes in the classification performance of different variants of the PIPE model, we use the ROC curve, which is a well-known machine learning tool for comparing binary classifiers (see section 2.1). The ROC curve allows us to compare classifiers even when the class ratio is not known *a priori*, and allows us to visualize the trade-off between false positive and false negative errors across the full range of specificities and sensitivities. The goal is to obtain ROC curves which are further northwest in the plot than the reference ROC curves. These curves represent classifiers

which can obtain higher sensitivity at the same specificity (north) or higher specificity at the same sensitivity (west).

To highlight the performance of the classifier at high specificity, we present each ROC curve as a pair of plots: one for the full range of specificities, and a zoomed-in version which focuses on the south-west quadrant of the plot (high specificity, generally between 100% and 90% specificity). Both plots are derived from the same experiment and the same pair scores. We are also interested, for comparison purposes, in a single numerical value which characterizes the classifier's performance. For this purpose, we report for each classifier its sensitivity at 99.95% specificity. This is the proportion of all positive instances which are discovered by the classifier as positive, while allowing only 0.05% of all negatives to be erroneously classified as positive. This provides a quantitative evaluation of the classifier as it might perform in an all-to-all screen, which requires very high specificity.

The sensitivity and specificity data points are obtained using leave-one-out cross-validation (LOOCV) to obtain unbiased estimates (see section 2.3.1). LOOCV is particular well-suited to the PIPE algorithm because excluding one pair from the training set is a relatively inexpensive operation. On the other hand, LOOCV allows us to use the entire training set as a sample for estimating classifier performance, producing very accurate estimates. As discussed in section 2.3.1, we use a variation on LOOCV specific to PIPE where, when testing a protein pair `(A, B)`, we exclude all pairs involving `A` and all pairs involving `B` from the known interaction database. This represents the case where `A` and `B` are novel proteins with no known interactions, and is the most stringent form of LOOCV as it applies to PIPE.

For each classifier to be evaluated, we perform a LOOCV and plot an ROC curve for both *S. cerevisiae* and *H. sapiens*. The *S. cerevisiae* set contains 43,591 positive pairs and 100,000 negative pairs and the *H. sapiens* set contains 41,678 positive pairs and 100,000 negative pairs. The positive pairs originate from BioGRID [61], which combines interactions from several other sources. Because negative protein interaction results are not typically reported, the negative pairs were generated randomly. We first chose a single protein at random, and then chose a partner at random (with replacement because self-interactions are valid). This pair was accepted in the negative set only if it wasn't already present in the positive set, but no additional constraints were placed on the pair before accepting it. As discussed in section 2.3, imposing such a constraint, such as requiring that the proteins are non-co-localized, has been shown to bias classifier accuracy estimates because the classifier learns the negative sampling constraint rather than the underlying basis for protein-protein interactions [40].

4.3 Results

4.3.1 Sequence Feature Hypotheses

First, we tested whether signal peptides and transmembrane regions, both experimental and predicted, are negatively correlated with binding sites in *S. cerevisiae*, with binding site data taken from DOMINO (Table 9). There were, in total, 225 proteins considered, which contained 162,824 amino acids. 13,581 of these amino acids were involved in binding sites. For each sequence feature, the number of amino acids which are both contained within a binding site and annotated with the given sequence feature is much lower than the expected count, if binding sites and those sequence features were uncorrelated (i.e. under the null hypothesis). Each hypothesis test rejected the

uncorrelated (i.e. null) hypothesis with a P-value of at most 3.5×10^{-4}. Our conclusion is that both signal peptide and transmembrane sequence features are negatively correlated with binding sites, and thus they are good predictors of the absence of a binding site in a particular protein region.

We similarly performed hypothesis tests for *S. cerevisiae* binding sites from PiSite (Table 10). This database contained binding site data for 370 proteins, totalling 163,567 amino acids. 19,638 of these were found to be involved in a binding site. We used the same 4 sources of sequence annotations as for the previous set of tests. Again, all independence hypotheses are rejected, at P-values of 8.2×10^{-3} or better. The experimental binding site data in PiSite is also well-predicted by both experimental and predicted signal peptide and transmembrane region features.

All amino acids in proteins with known binding sites	162,824	162,824	162,824	162,824
Amino acids in binding sites	13,581	13,581	13,581	13,581
Sequence feature	Experimental signal peptides	Predicted signal peptides	Experimental transmembrane	Predicted transmembrane
Annotated amino acids	144	365	3,135	2,944
Amino acids both in binding site and annotated	2	2	5	3
Expected if events were independent	12	20	261	245
Independence hypothesis	Rejected	Rejected	Rejected	Rejected
P-value	3.5×10^{-4}	8.8×10^{-12}	2.8×10^{-110}	1.3×10^{-106}

Table 9: Results of hypothesis tests for negative correlation between *S. cerevisiae* binding sites from DOMINO and experimental/predicted signal peptides/transmembrane regions.

All amino acids in proteins with known binding sites	163,567	163,567	163,567	163,567
Amino acids in binding sites	19,638	19,638	19,638	19,638
Sequence feature	Experimental signal peptides	Predicted signal peptides	Experimental transmembrane	Predicted transmembrane
Annotated amino acids	79	274	808	542
Amino acids both in binding site and annotated	0	13	75	25
Expected if events were independent	9	32	97	65
Independence hypothesis	Rejected	Rejected	Rejected	Rejected
P-value	4.1×10^{-5}	3.1×10^{-5}	8.2×10^{-3}	2.2×10^{-9}

Table 10: Hypothesis tests for negative correlation between *S. cerevisiae* PiSite binding sites and sequence features.

We repeated the experiments to test whether signal peptides and transmembrane regions were also negatively correlated with binding sites in *H. sapiens* proteins. From DOMINO, we extracted binding site data for 649 proteins, totalling 560,877 amino acids, 70,384 of which were in a binding site. We tested whether experimental and predicted signal peptides and transmembrane annotates predicted these binding site data (Table 11), and found that the independence hypotheses are again rejected for all types of sequence feature data. Sequence feature data also predicts binding sites derived from PiSite (Table 12), where we examined 1,648 proteins totalling 874,331 amino acids, 76,775 of which were involved in a binding site.

All amino acids in proteins with known binding sites	560,877	560,877	560,877	560,877
Amino acids in binding sites	70,384	70,384	70,384	70,384
Sequence feature	Experimental signal peptides	Predicted signal peptides	Experimental transmembrane	Predicted transmembrane
Annotated amino acids	2,970	3,270	9,378	9,346
Amino acids both in binding site and annotated	107	99	194	200
Expected if events were independent	372	410	1,176	1,172
Independence hypothesis	Rejected	Rejected	Rejected	Rejected
P-value	2.2×10^{-35}	1.5×10^{-83}	3.2×10^{-305}	6.0×10^{-109}

Table 11: Hypothesis tests for negative correlation between *H. sapiens* DOMINO binding sites and sequence features.

All amino acids in proteins with known binding sites	874,331	874,331	874,331	874,331
Amino acids in binding sites	76,775	76,775	76,775	76,775
Sequence feature	Experimental signal peptides	Predicted signal peptides	Experimental transmembrane	Predicted transmembrane
Annotated amino acids	9,860	10,934	8,658	9,950
Amino acids both in binding site and annotated	20	92	145	153
Expected if events were independent	865	960	760	873
Independence hypothesis	Rejected	Rejected	Rejected	Rejected
P-value	$< 10^{-308}$	1.1×10^{-303}	2.6×10^{-176}	4.2×10^{-213}

Table 12: Hypothesis tests for negative correlation between *H. sapiens* PiSite binding sites and sequence features.

4.3.2 Effect of Functional Annotations

Having validated our hypotheses about signal peptides and transmembrane being negative predictors of binding sites, we integrated this additional data into the PIPE model (see section 4.2.2) and compared the classification performance between the original model and the model augmented with sequence feature data. We performed separate experiments for *S. cerevisiae* and *H. sapiens* and for each source of sequence annotations: experimental signal peptide data, predicted signal peptides, experimental transmembrane regions, and predicted transmembrane regions. For each organism, we also constructed a combined model in which both signal peptides and transmembrane regions are masked out, in order to draw on both sources of additional data.

For brevity in the text, we compare classifiers only by the sensitivity at the 99.95% specificity point in the ROC curve of each classifier. The full ROC curves are given in Appendix A, where the unmodified PIPE model is labeled "Original" and each modified model which integrates sequence annotation data is plotted alongside the first curve for comparison. There are in total 4 ROC curve comparison plots in *S. cerevisiae* (section 7.1) and 4 in *H. sapiens* (section 7.2).

The sensitivities at 99.95% specificity show that integrating sequence annotations into the model is only sometimes beneficial (Table 13). In *S. cerevisiae*, signal peptide data has no noticeable effect. This can be attributed to the fact that in this organism, signal peptides represent a very small percentage of the amino acids in a sample of proteins (Table 9) in both DOMINO and PiSite proteins (1.1% and 0.4%). It is particularly interesting that the larger sample of proteins (PiSite), has the smaller percentage of amino acids involved in signal peptides. On the other hand, in *H. sapiens* where signal peptides represent 4.2% of amino acids in DOMINO and 12.8% of amino acids in PiSite, incorporating signal peptide data provides a noticeable increase in classifier performance (1.26% to 4.23%).

Experimental transmembrane data only slightly improves *S. cerevisiae* performance, while predicted transmembrane data in fact hinders classification performance. A comparable number of amino acids are annotated as transmembrane in both cases (3,135 experimental versus 2,944 predicted), so this reduced performance could be attributed to errors in the predicted *location* of the transmembrane regions, or insufficient representation of *S. cerevisiae* in the training set (only 160 proteins [56]). On the other hand, transmembrane data improves classification performance in *H. sapiens*, from 1.26% to ~3.6% sensitivity, regardless of the source of the transmembrane annotation.

The performance of the combined models is dominated by the transmembrane region modifications, which cover the largest proportion of amino acids in each respective proteome. These cover approximately 23% of amino acids in *S. cerevisiae* and 13% of amino acids in *H. sapiens*. This means that for *S. cerevisiae* the overall best

performances is achieved with the model which incorporates only transmembrane region data, while in *H. sapiens* the best performing model is the signal peptide-only model.

Organism	Model	Sensitivity at 99.95% Specificity
S. cerevisiae	Original	2.17%
	Using experimental signal peptides	2.17%
	Using predicted signal peptides	2.16%
	Using experimental transmembrane regions	2.25%
	Using predicted transmembrane regions	1.76%
	Combined experimental signal peptide and transmembrane data	2.23%
H. sapiens	Original	1.26%
	Using experimental signal peptides	4.23%
	Using predicted signal peptides	4.18%
	Using experimental transmembrane regions	3.64%
	Using predicted transmembrane regions	3.61%
	Combined experimental signal peptide and transmembrane data	3.65%

Table 13: Performance of classifiers which incorporate sequence feature data in *S. cerevisiae* and *H. sapiens*

4.3.3 Effect of Sequence Window Uniqueness

We modified the PIPE algorithm to account for sequence window uniqueness, as described in section 4.2.3. For each cell of the output matrix, after counting the number of hits in the known interactions database, we additionally divide this value by the total possible number of interaction hits, which is the product of the number of similarity hits of the two query windows. When averaging the matrix to obtain an overall pair score, this reduces the score inflation due to highly re-occurring sequence regions, while enhancing the contribution of highly unique sequence windows.

Both in *S. cerevisiae* and in *H. sapiens*, the modifications result in a noticeable improvement of the sensitivity at 99.95% specificity (Table 14). The sensitivity increases from 2.17% to 3.05% in *S. cerevisiae* and from 1.26% to 16.19% in *H. sapiens*. The ROC curves for the *S. cerevisiae* classifier (Figure 18) and the *H. sapiens* classifier (Figure 19) confirm that the improvement in sensitivity is observed across the entire operating range

of specificities, though the relative improvement decreases at lower specificities. This model exhibits the best performance out of all models evaluated here, including all variants of the sequence annotation models. This is particularly useful, since this model requires no additional data other than what is already available in the PIPE database, and thus is not bottlenecked by the availability of other protein sequence annotation data.

Organism	Model	Sensitivity at 99.95% Specificity
S. cerevisiae	Original	2.17%
	Using sequence window uniqueness	3.05%
H. sapiens	Original	1.26%
	Using sequence window uniqueness	16.19%

Table 14: Performance of classifiers which incorporate sequence window uniqueness in *S. cerevisiae* and *H. sapiens*

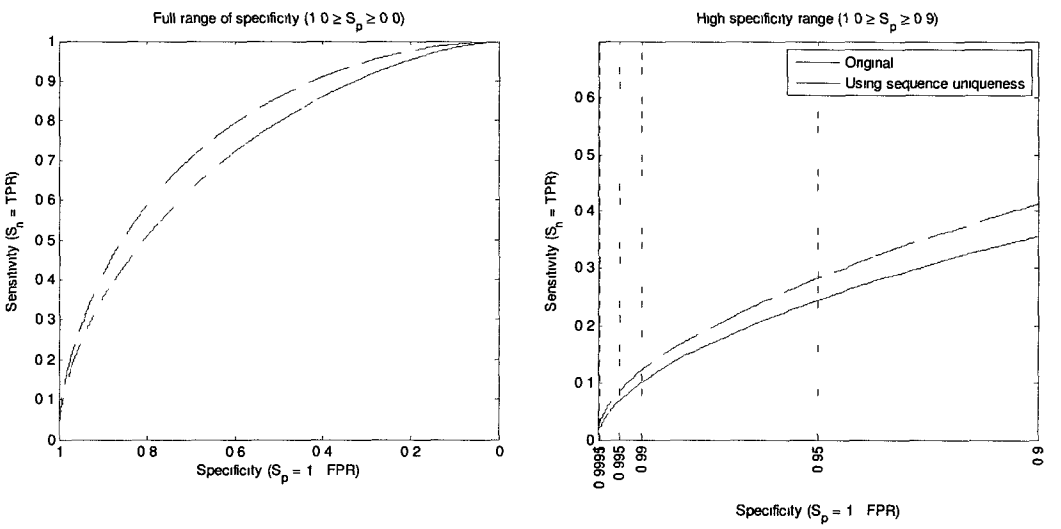


Figure 18: ROC curves of original model and model which incorporates sequence window uniqueness in *S. cerevisiae*

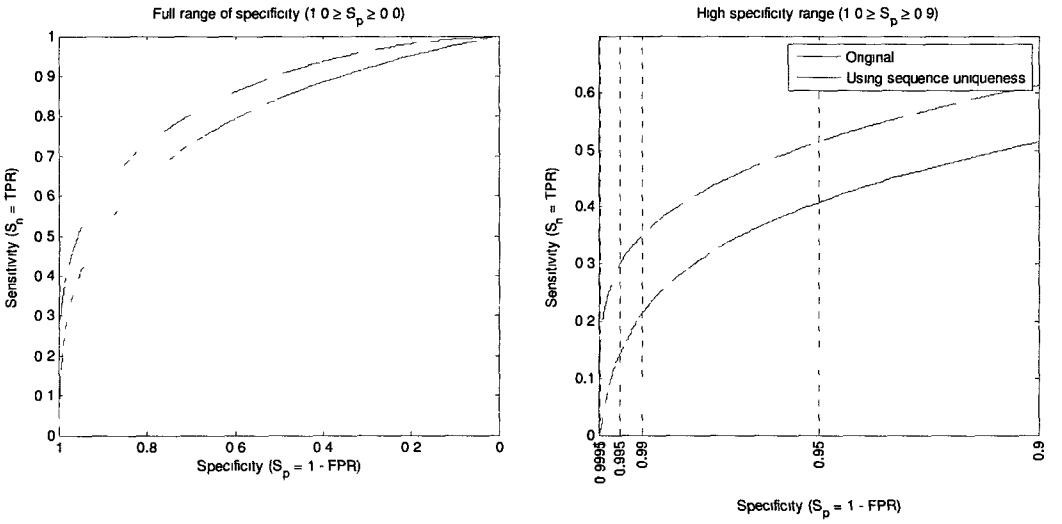


Figure 19: ROC curves of original model and model which incorporates sequence window uniqueness in *H. sapiens*

We did not construct classifiers which combined signal peptide and transmembrane annotations with sequence window uniqueness. Since signal peptides and transmembrane regions re-occur throughout the proteome, we expect that they are already accounted for by sequence window uniqueness as non-unique regions, thus reducing their score contribution. Masking these sequence regions prior to classification by the model with sequence uniqueness would not have a large additional effect compared to sequence uniqueness alone.

Accounting for sequence window uniqueness requires additional processing, which has the potential to impact the runtime of the algorithm. However, in practice, the additional time required for a validation run, compared to the original model, was on the order of minutes. This is most likely due to the small number of additional operations required to compute the new output score (ratio from Figure 17) compared to the processing required to obtain the number of interaction hits (numerator). Thus, the

classification performance of PIPE is improved while preserving its computational performance.

The increased computational speed of the algorithm opens the door to experimentation with more complex ways of computing the output score. For example, if we assume that interaction hits are independent random events, then the total number of hits follows a binomial distribution and the output score can be computed as a P-value of observing a particular number of hits. This may further improve the sensitivity of the classifier, though this has not been tested. A future study could explore this and other methods of calculating this output score.

4.4 Conclusion

We have proposed three modified PIPE models which either take into account additional protein sequence data, or which make better use of the existing data in the PIPE database. In general, all models attempt to account for highly re-occurring sequence regions which are not involved in PPIs. The last model performed best on both *S. cerevisiae* and *H. sapiens* validation sets. In a high-specificity all-to-all screen, we estimate that this new model would be able to detect 3.05% of the true interactions in the *S. cerevisiae* protein interaction network (41% more than the original model) and 16.19% of the true interactions in the *H. sapiens* protein interaction network (1,185% more than the original model).

5 Conclusion

In Chapter 1, we introduced the field of protein-protein interaction (PPI) prediction and stated the goals of this thesis. Chapter 2 defined the pattern classification framework used to quantify the performance of PPI classifiers and compared several commonly-cited PPI predictors from literature, identifying one method which was the state of the art in its field (PIPE). In Chapters 3 and 4, we proposed changes to the implementation and model of PIPE which improve its computational and classification performance. Finally, in this chapter, we summarize our conclusions and provide avenues for further potential improvement of PIPE.

5.1 Conclusions

First, we compared 6 commonly-cited methods for PPI prediction [34] [11] [35] [36] [37] [17]. Wherever possible, we attempted to compensate for the biases of the authors' chosen validation sets and performance metrics by converting their stated results into unbiased measured of classifier performance. The last method, the Protein-protein Interaction Prediction Engine (PIPE), was shown to achieve the highest specificity, which meant that it was applicable to organism-wide PPI screens, thus representing the state of the art in the field. The computational performance of the method was potentially a limiting factor, and the classification performance was shown to have room for improvement [15].

Our first contribution is the improvement of the computational performance of PIPE. By restructuring the input data to the algorithm, and by using a more efficient implementation in a commonly-executed fragment of the algorithm, we greatly reduced

the average execution time per protein pair. Our test sets of 43,591 *S. cerevisiae* positive pairs, 100,000 *S. cerevisiae* negative pairs, 41,678 *H. sapiens* positive pairs and 100,000 *H. sapiens* negative pairs executed 8.1 , 8.9, 4.0 and 14.5 times faster, respectively, than the original algorithm. An interaction screen between all protein pairs in *H. sapiens* would now execute on a 28-core cluster in 20 days with the proposed changes, instead of 9 months with the original implementation.

Second, we improved the classification performance of PIPE. First, we showed that certain types of sequence features (signal peptides and transmembrane regions) are negatively correlated with protein regions involved in PPIs. Integrating this data into the PIPE model resulted in an improvement of sensitivity from 2.17% to 2.25% in *S. cerevisiae* protein pairs with experimental transmembrane data, and from 1.26% to 4.23% in *H. sapiens* protein pairs with experimental signal peptide data. However, a generalization of this concept, which uses data from PIPE's own sequence comparison database to reduce the weight given to highly re-occurring sequence regions, resulted in much larger gains in classification performance. At 99.95% specificity, which is suitable for organism-wide screens, the specificity in *S. cerevisiae* increased from 2.17% to 3.06%; in *H. sapiens*, the sensitivity increased from 1.26% to 16.19%. If PIPE were to classify every protein pair in the human proteome, we estimate that only 0.05% of results would be false positive and 16.19% of the true interactions would be discovered. Previously published methods trade in lower specificity (typically 99% or less) for higher sensitivity, which makes them unusable for organism-wide screens.

Together, these improvements greatly increase the utility of PIPE as a tool for the study of organism-wide protein interaction networks. Wet lab techniques provide the

much needed training data for PPI classifiers, but cannot keep up with the demand for ever-greater coverage of the proteome. Lastly, computational methods benefit from the continuous increase in performance and decrease in cost of new generations of computer hardware.

5.2 Future Work

Several aspects of the PIPE model and implementation remain to be studied in more detail, and could lead to further improved computational and/or classification performance.

5.2.1 Computational Performance

First, the results for algorithm runtime presented in section 3.5 were obtained using single measurements of runtime per protein pair. This is a shortcoming of the benchmarking methodology because, as discussed in section 3.4, noise from various sources could have interfered with the measurements. A future study could verify that the performance improvements are consistent across repeated measurements, which would reduce the impact of measurement noise.

The original runtime for a given pair is not the only possible control variable to be considered when analyzing the effect of the algorithm and implementation modifications. Other protein and protein pair properties, such as protein length, or the number of known interactions for a given protein, could better account for the variability in speedups observed. The PIPE score given to a protein pair could also be a useful control variable. In particular, for *H. sapiens* protein pairs, original runtime seems to have been insufficient as the sole variable for very long-running pairs. By better predicting the impact of certain optimizations to specific protein pairs based on their properties, one can

selectively apply optimizations in order to maximize the benefit for specific classes of protein pairs.

The neighbour list is a very frequently-accessed data structure of the PIPE algorithm. The storage format proposed in section 3.2.3 (arrays padded to the maximum length) makes much better use of the CPU cache than the previous format (linked lists), but may benefit from further optimization. In particular, very short protein ID arrays, which occupy less than one cache line, make inefficient use of the cache. A special case for these very short protein ID arrays may improve the performance of accesses into the neighbour list as a whole, depending on the distribution of the lengths of neighbour lists for the organism under consideration. Any attempted change to this storage format must be careful not to render the algorithm CPU-bound in exchange for a marginal improvement in cache hit rate, which would penalize the case where cache hit rate is already very high.

In general, the broad-stroke optimizations presented here can be furthered by using a cache profiling tool (e.g. Valgrind's sub-tool Cachegrind [62]) and quantitatively measuring cache hit rates for various parts of the algorithm, and for specific changes to the implementation. Care should be taken to quantify the effect of algorithm changes for a representative set of protein pairs. For this level of analysis, the separation of protein pairs into positive and negative pairs is most likely not sufficient to represent the wide variety of protein pairs, with different memory access patterns, encountered in an organism-wide screen.

5.2.2 Classification Performance

Signal peptides and transmembrane regions were shown to be negatively correlated with PPI-mediating protein regions (section 4.2.1). DNA-binding motifs, and other well-annotated sequence features, may also be negatively correlated. A subsequent study might test analogous hypotheses for many different types of sequence features.

Weighing the number of hits in the known interactions database based on the uniqueness of the sequence windows under consideration was shown to improve classification performance (section 4.2.3). We used three variables to generate the final score: the number of hits in the known interactions database (knownPairsHits), the number of similarity hits for the current sequence window from protein A ($simHitsA_i$), and the number of similarity hits for the current sequence window from protein B ($simHitsB_j$). The final score formula was: $H_{i,j} = \frac{knownPairHits}{simHitsA_i * simHitsB_j}$. This is only one of many possible functions $H_{i,j} = H(knownPairHits, simHitsA_i, simHitsB_j)$. For example, there may be a limiting level of similarity hits $simHitsA_i$ (or $simHitsB_j$, by symmetry) past which knownPairHits should no longer be down-weighted. Another possibility is to substitute for H the statistical significance of finding a particular value of knownPairHits, given some hypothesized prior distribution for knownPairHits. Unfortunately, measuring the impact on classification performance of a trial H function is costly because it requires an entire leave-one-out experiment. This penalty has been slightly reduced as a result of our performance improvements presented in Chapter 3, but remains a significant barrier to model tweaking.

Similarly, instead of aggregating the PIPE matrix (landscape) values by calculating the mean, there may be other aggregation functions which produce better-

discriminating pair scores. One could also treat the PIPE matrix as a 2-D image and apply a filter before calculating the aggregate value. This approach was used in the second revision of PIPE [17], where it was found to improve classification performance. Care should be taken that the filter be computationally-efficient enough to be worth the improvement in classification performance.

Finally, there may be additional information to be gained from the PAM scores used to compare sequence windows for similarity. Currently, a threshold is applied to the PAM score and the result is used as a binary feature – two sequence windows are either similar or not. There may be additional information to be gained by incorporating the degree of similarity between two amino acid windows into the calculation of the final pair score. Here too, computational efficiency must be carefully considered. Precomputing and storing the PAM scores of all pairs of sequence windows is space-prohibitive, while computing these PAM scores on the fly is prohibitive in terms of computation time [16].

6 Bibliography

[1] F. Chiti and C. M. Dobson, "Protein Misfolding, Functional Amyloid, and Human Disease," *Annual Review of Biochemistry*, vol. 75, no. 1, pp. 333-366, Jul. 2006.

[2] P. D. Kwong, R. Wyatt, J. Robinson, R. W. Sweet, J. Sodroski, and W. A. Hendrickson, "Structure of an HIV gp120 envelope glycoprotein in complex with the CD4 receptor and a neutralizing human antibody.," *Nature*, vol. 393, no. 6686, pp. 648-59, Jun. 1998.

[3] D. J. Kempf, K. C. Marsh, J. F. Denissen, E. McDonald, S. Vasavanonda, C. A. Flentge, B. E. Green, L. Fino, C. H. Park, and X. P. Kong, "ABT-538 is a potent inhibitor of human immunodeficiency virus protease and has high oral bioavailability in humans.," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 92, no. 7, pp. 2484-8, Mar. 1995.

[4] L. D. Stein, "Human genome: end of the beginning.," *Nature*, vol. 431, no. 7011, pp. 915-6, Oct. 2004.

[5] L. W. Hillier, A. Coulson, J. I. Murray, Z. Bao, J. E. Sulston, and R. H. Waterston, "Genomics in C. elegans: so many genes, such a little worm.," *Genome Research*, vol. 15, no. 12, pp. 1651-60, Dec. 2005.

[6] S. A. Goff, D. Ricke, T.-H. Lan, G. Presting, R. Wang, M. Dunn, J. Glazebrook, A. Sessions, P. Oeller, H. Varma, D. Hadley, D. Hutchison, et al., "A draft sequence of the rice genome (Oryza sativa L. ssp. japonica).," *Science*, vol. 296, no. 5565, pp. 92-100, May 2002.

[7] M. P. H. Stumpf, T. Thorne, E. de Silva, R. Stewart, H. J. An, M. Lappe, and C. Wiuf, "Estimating the size of the human interactome.," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 105, no. 19, pp. 6959-64, May 2008.

[8] N. Tuncbag, G. Kar, O. Keskin, A. Gursoy, and R. Nussinov, "A survey of available tools and web servers for analysis of protein-protein interactions and interfaces.," *Briefings in Bioinformatics*, vol. 10, no. 3, pp. 217-32, May 2009.

[9] C. von Mering, R. Krause, B. Snel, M. Cornell, S. G. Oliver, S. Fields, and P. Bork, "Comparative assessment of large-scale data sets of protein-protein interactions.," *Nature*, vol. 417, no. 6887, pp. 399-403, May 2002.

[10] H. M. Berman, "The Protein Data Bank," *Nucleic Acids Research*, vol. 28, no. 1, pp. 235-242, Jan. 2000.

[11] J. R. Bock and D. A. Gough, "Predicting protein-protein interactions from primary structure," *Bioinformatics*, vol. 17, no. 5, pp. 455-460, May 2001.

[12] J. C. Venter, M. D. Adams, E. W. Myers, P. W. Li, R. J. Mural, G. G. Sutton, H. O. Smith, M. Yandell, C. A. Evans, R. A. Holt, J. D. Gocayne, P. Amanatides, et al., "The sequence of the human genome.," *Science*, vol. 291, no. 5507, pp. 1304-51, Feb. 2001.

[13] E. S. Lander, L. M. Linton, B. Birren, C. Nusbaum, M. C. Zody, J. Baldwin, K. Devon, K. Dewar, M. Doyle, W. FitzHugh, R. Funke, D. Gage, et al., "Initial sequencing and analysis of the human genome.," *Nature*, vol. 409, no. 6822, pp. 860-921, Feb. 2001.

[14] K. Liolios, I.-M. A. Chen, K. Mavromatis, N. Tavernarakis, P. Hugenholtz, V. M. Markowitz, and N. C. Kyrpides, "The Genomes On Line Database (GOLD) in 2009: status of genomic and metagenomic projects and their associated metadata.," *Nucleic Acids Research*, vol. 38, no. Database issue, pp. D346-54, Jan. 2010.

[15] Y. Park, "Critical assessment of sequence-based protein-protein interaction prediction methods that do not require homologous protein sequences," *BMC Bioinformatics*, vol. 10, p. 419, Jan. 2009.

[16] S. Pitre, F. Dehne, A. Chan, J. Cheetham, A. Duong, A. Emili, M. Gebbia, J. Greenblatt, M. Jessulat, N. Krogan, X. Luo, and A. Golshani, "PIPE: a protein-protein interaction prediction engine based on the re-occurring short polypeptide sequences between known interacting protein pairs.," *BMC Bioinformatics*, vol. 7, no. 1, p. 365, Jan. 2006.

[17] S. Pitre, C. North, M. Alamgir, M. Jessulat, A. Chan, X. Luo, J. R. Green, M. Dumontier, F. Dehne, and A. Golshani, "Global investigation of protein-protein interactions in yeast Saccharomyces cerevisiae using re-occurring short polypeptide sequences.," *Nucleic Acids Research*, vol. 36, no. 13, pp. 4286-94, Aug. 2008.

[18] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861-874, Jun. 2006.

[19] S. Jones and J. M. Thornton, "Principles of protein-protein interactions.," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 93, no. 1, pp. 13-20, Jan. 1996.

[20] S. Jones and J. M. Thornton, "Prediction of protein-protein interaction sites using patch analysis.," *Journal of Molecular Biology*, vol. 272, no. 1, pp. 133-43, Sep. 1997.

[21] P. Fariselli, F. Pazos, A. Valencia, and R. Casadio, "Prediction of protein-protein interaction sites in heterocomplexes with neural networks," *European Journal of Biochemistry*, vol. 269, no. 5, pp. 1356-1361, Mar. 2002.

[22] Y. Murakami and S. Jones, "SHARP2: protein-protein interaction predictions using patch analysis.," *Bioinformatics*, vol. 22, no. 14, pp. 1794-5, Jul. 2006.

[23] T. Kortemme and D. Baker, "A simple physical model for binding energy hot spots in protein-protein complexes," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 99, no. 22, pp. 14116-21, Oct. 2002.

[24] F. P. Davis, H. Braberg, M.-Y. Shen, U. Pieper, A. Sali, and M. S. Madhusudhan, "Protein complex compositions predicted by structural similarity.," *Nucleic Acids Research*, vol. 34, no. 10, pp. 2943-52, Jan. 2006.

[25] S. H. Jung, B. Hyun, W.-H. Jang, H.-Y. Hur, and D.-S. Han, "Protein complex prediction based on simultaneous protein interaction network.," *Bioinformatics*, vol. 26, no. 3, pp. 385-91, Feb. 2010.

[26] R. D. Finn, J. Mistry, J. Tate, P. Coggill, A. Heger, J. E. Pollington, O. L. Gavin, P. Gunasekaran, G. Ceric, K. Forslund, L. Holm, E. L. L. Sonnhammer, et al., "The Pfam protein families database.," *Nucleic Acids Research*, vol. 38, no. Database issue, pp. D211-22, Jan. 2010.

[27] S. Hunter, R. Apweiler, T. K. Attwood, A. Bairoch, A. Bateman, D. Binns, P. Bork, U. Das, L. Daugherty, L. Duquenne, R. D. Finn, J. Gough, et al., "InterPro: the integrative protein signature database.," *Nucleic Acids Research*, vol. 37, no. Database issue, pp. D211-5, Jan. 2009.

[28] R. Apweiler, "Ongoing and future developments at the Universal Protein Resource," *Nucleic Acids Research*, Nov. 2010.

[29] R. M. Kini and H. J. Evans, "A hypothetical structural role for proline residues in the flanking segments of protein-protein interaction sites.," *Biochemical and Biophysical Research Communications*, vol. 212, no. 3, pp. 1115-24, Jul. 1995.

[30] X. Gallet, B. Charloteaux, A. Thomas, and R. Brasseur, "A fast method to predict protein interaction sites from sequences.," *Journal of Molecular Biology*, vol. 302, no. 4, pp. 917-26, Sep. 2000.

[31] Y. Ofran and B. Rost, "Analysing Six Types of Protein–Protein Interfaces," *Journal of Molecular Biology*, vol. 325, no. 2, pp. 377-387, Jan. 2003.

[32] Y. Ofran and B. Rost, "Predicted protein–protein interaction sites from local sequence information," *FEBS Letters*, vol. 544, no. 1-3, pp. 236-239, Jun. 2003.

[33] B. Ma, T. Elkayam, H. Wolfson, and R. Nussinov, "Protein-protein interactions: structurally conserved residues distinguish between binding sites and exposed protein surfaces.," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 100, no. 10, pp. 5772-7, May 2003.

[34] E. Sprinzak and H. Margalit, "Correlated sequence-signatures as markers of protein-protein interaction.," *Journal of Molecular Biology*, vol. 311, no. 4, pp. 681-92, Aug. 2001.

[35] S. Martin, D. Roe, and J.-L. Faulon, "Predicting protein-protein interactions using signature products," *Bioinformatics*, vol. 21, no. 2, pp. 218-26, Jan. 2005.

[36] J. Shen, J. Zhang, X. Luo, W. Zhu, K. Yu, K. Chen, Y. Li, and H. Jiang, "Predicting protein-protein interactions based only on sequences information.," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 104, no. 11, pp. 4337-41, Mar. 2007.

[37] Y. Guo, L. Yu, Z. Wen, and M. Li, "Using support vector machine combined with auto covariance to predict protein-protein interactions from protein sequences.," *Nucleic Acids Research*, vol. 36, no. 9, pp. 3025-30, May 2008.

[38] S. Wold, J. Jonsson, M. Sjörström, M. Sandberg, and S. Rännar, "DNA and peptide sequences and chemical processes multivariately modelled by principal component analysis and partial least-squares projections to latent structures," *Analytica Chimica Acta*, vol. 277, no. 2, pp. 239-253, May 1993.

[39] M. V. Heel, "A new family of powerful multivariate statistical sequence analysis techniques," *Journal of Molecular Biology*, vol. 220, no. 4, pp. 877-887, Aug. 1991.

[40] A. Ben-Hur and W. S. Noble, "Choosing negative examples for the prediction of protein-protein interactions.," *BMC Bioinformatics*, vol. 7 Suppl 1, p. S2, Jan. 2006.

[41] M. O. Dayhoff and R. M. Schwartz, "A Model of Evolutionary Change in Proteins," in *Atlas of Protein Sequence and Structure (Vol 5, Supplement 3)*, National Biomedical Research Foundation, 1979, pp. 345-352.

[42] A. Stein and P. Aloy, "Novel Peptide-Mediated Interactions Derived from High-Resolution 3-Dimensional Structures," *PLoS Computational Biology*, vol. 6, no. 5, p. e1000789, May 2010.

[43] T. S. Keshava Prasad, R. Goel, K. Kandasamy, S. Keerthikumar, S. Kumar, S. Mathivanan, D. Telikicherla, R. Raju, B. Shafreen, A. Venugopal, L. Balakrishnan, A. Marimuthu, et al., "Human Protein Reference Database--2009

update.," *Nucleic Acids Research*, vol. 37, no. Database issue, pp. D767-72, Jan. 2009.

[44] C.-Y. Yu, L.-C. Chou, and D. T.-H. Chang, "Predicting protein-protein interactions in unbalanced data using the primary structure of proteins.," *BMC Bioinformatics*, vol. 11, p. 167, Jan. 2010.

[45] H. W. Meuer, "The TOP500 Project: Looking Back Over 15 Years of Supercomputing Experience," *Informatik-Spektrum*, vol. 31, no. 3, pp. 203-222, Apr. 2008.

[46] R. A. Baeza-Yates and A. Salinger, "Experimental Analysis of a Fast Intersection Algorithm for Sorted Sequences," in *SPIRE*, 2005, pp. 13-24.

[47] J. S. Culpepper and A. Moffat, "Compact set representation for information retrieval," in *Proceedings of the 14th international conference on String processing and information retrieval*, 2007, pp. 137-148.

[48] D. Knuth, "Section 5.2.4: Sorting by Merging," in *The Art of Computer Programming*, 2nd ed., Addison-Wesley, 1998, pp. 158-168.

[49] P. R. Cohen, *Empirical Methods for Artificial Intelligence*, First Edit. Cambridge, MA: MIT Press, 1995.

[50] C. J. Morton and I. D. Campbell, "SH3 Domains: Molecular 'Velcro'," *Current Biology*, vol. 4, no. 7, pp. 615-617, Jul. 1994.

[51] M. Paetzel, A. Karla, N. C. J. Strynadka, and R. E. Dalbey, "Signal Peptidases," *Chemical Reviews*, vol. 102, no. 12, pp. 4549-4580, Dec. 2002.

[52] S. R. Goodman, "Cell Membranes," in *Medical Cell Biology*, 3rd ed., S. R. Goodman, Ed. Academic Press, 2008, pp. 27-58.

[53] J. D. Bendtsen, H. Nielsen, G. von Heijne, and S. Brunak, "Improved prediction of signal peptides: SignalP 3.0.," *Journal of Molecular Biology*, vol. 340, no. 4, pp. 783-95, Jul. 2004.

[54] O. Emanuelsson, S. Brunak, G. von Heijne, and H. Nielsen, "Locating proteins in the cell using TargetP, SignalP and related tools.," *Nature Protocols*, vol. 2, no. 4, pp. 953-71, Jan. 2007.

[55] E. L. Sonnhammer, G. von Heijne, and A. Krogh, "A hidden Markov model for predicting transmembrane helices in protein sequences.," *Proceedings of the International Conference on Intelligent Systems for Molecular Biology*, vol. 6, pp. 175-82, Jan. 1998.

[56] A. Krogh, B. Larsson, G. von Heijne, and E. L. Sonnhammer, "Predicting transmembrane protein topology with a hidden Markov model: application to complete genomes.," *Journal of Molecular Biology*, vol. 305, no. 3, pp. 567-80, Jan. 2001.

[57] A. Ceol, A. Chatr-aryamontri, E. Santonico, R. Sacco, L. Castagnoli, and G. Cesareni, "DOMINO: a database of domain-peptide interactions," *Nucleic Acids Research*, vol. 35, no. Database issue, pp. D557-60, Jan. 2007.

[58] A. Amos-Binks, "Proteome-Scale Protein-Protein Interaction Site Prediction and Novel Motif Discovery Using Re-Occurring Polypeptide Sequences," Carleton University, 2009.

[59] M. Higurashi, T. Ishida, and K. Kinoshita, "PiSite: a database of protein interaction sites using multiple binding states in the PDB.," *Nucleic Acids Research*, vol. 37, no. Database issue, pp. D360-4, Jan. 2009.

[60] A. Agresti, "A Survey of Exact Inference for Contingency Tables," *Statistical Science*, vol. 7, no. 1, pp. pp. 131-153, 1992.

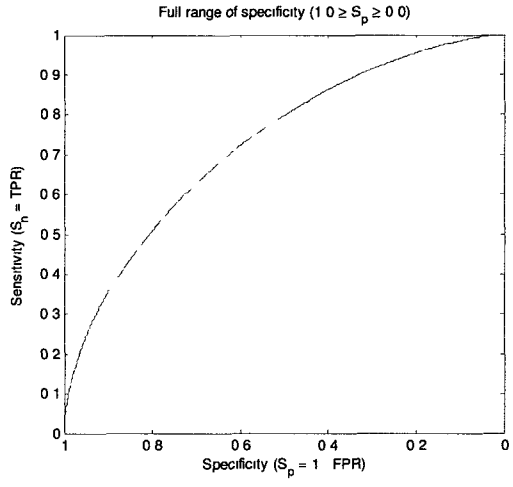
[61] B. Breitkreutz, C. Stark, T. Reguly, L. Boucher, A. Breitkreutz, M. Livstone, R. Oughtred, D. H. Lackner, J. Bähler, V. Wood, K. Dolinski, and M. Tyers, "The BioGRID Interaction Database: 2008 update.," *Nucleic Acids Research*, vol. 36, no. Database issue, pp. D637-40, Jan. 2008.

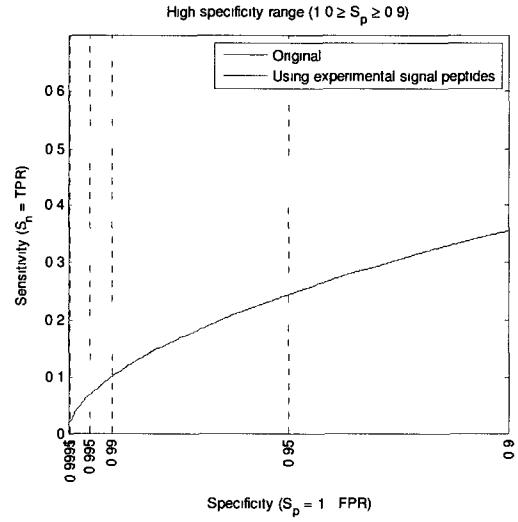
[62] N. Nethercote, "Dynamic Binary Analysis and Instrumentation," University of Cambridge, 2004.

7 Appendix A: ROC Curves for Classifiers Evaluated in Section 4.3.2

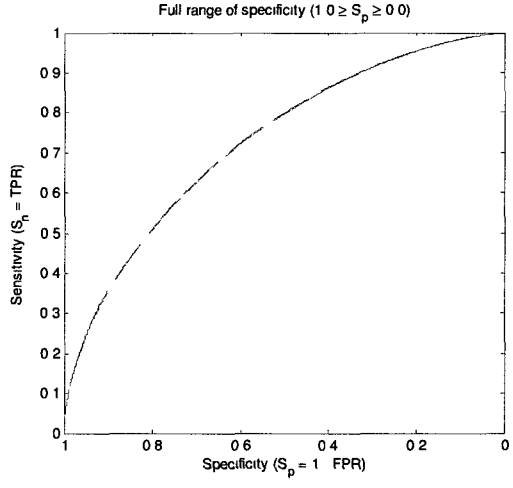
7.1 S. cerevisiae

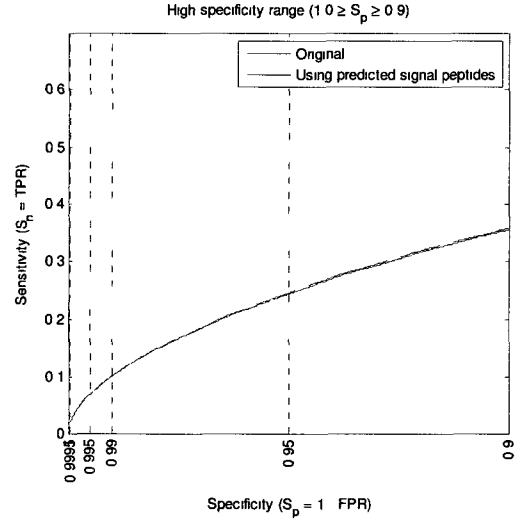
Model using experimental signal peptide data



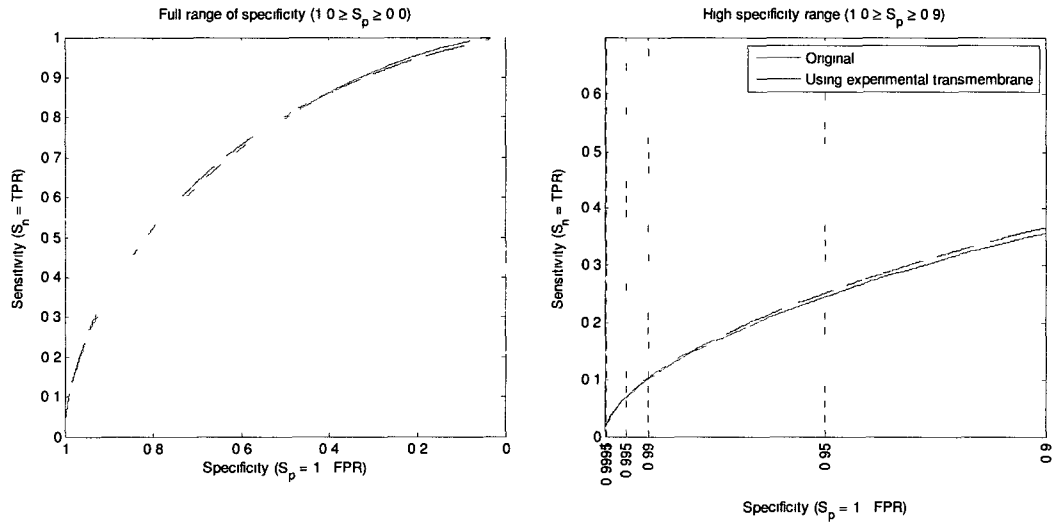


Model using predicted signal peptide data

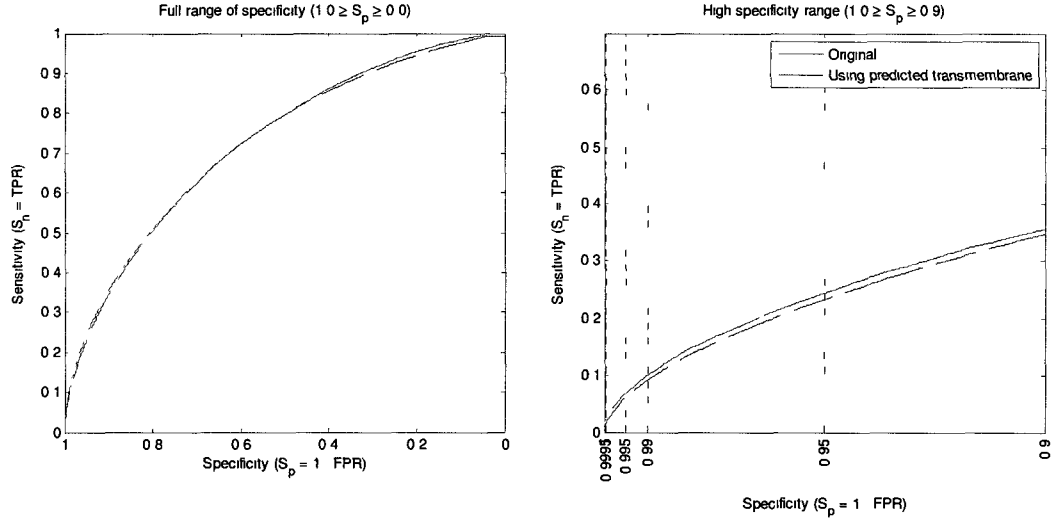




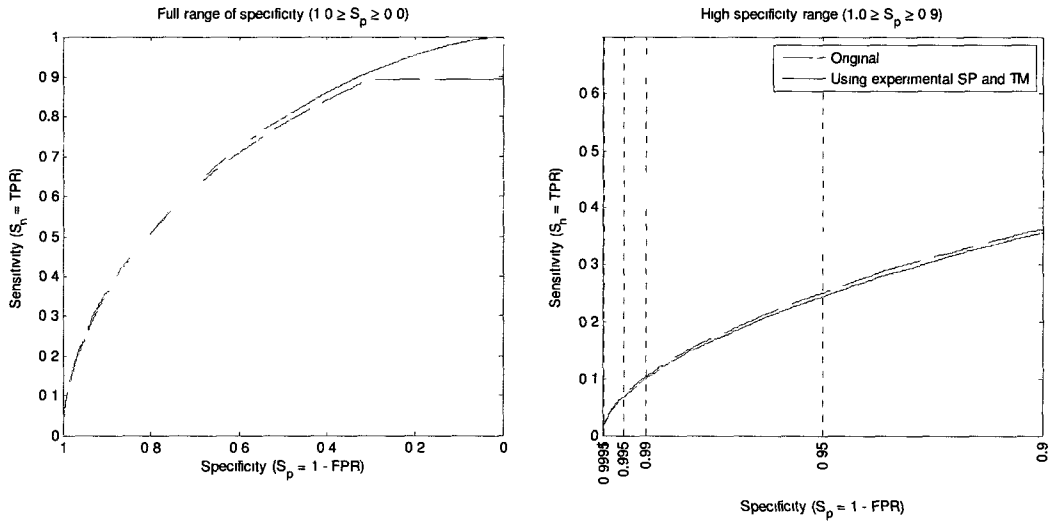
Model using experimental transmembrane region data



Model using predicted transmembrane region data

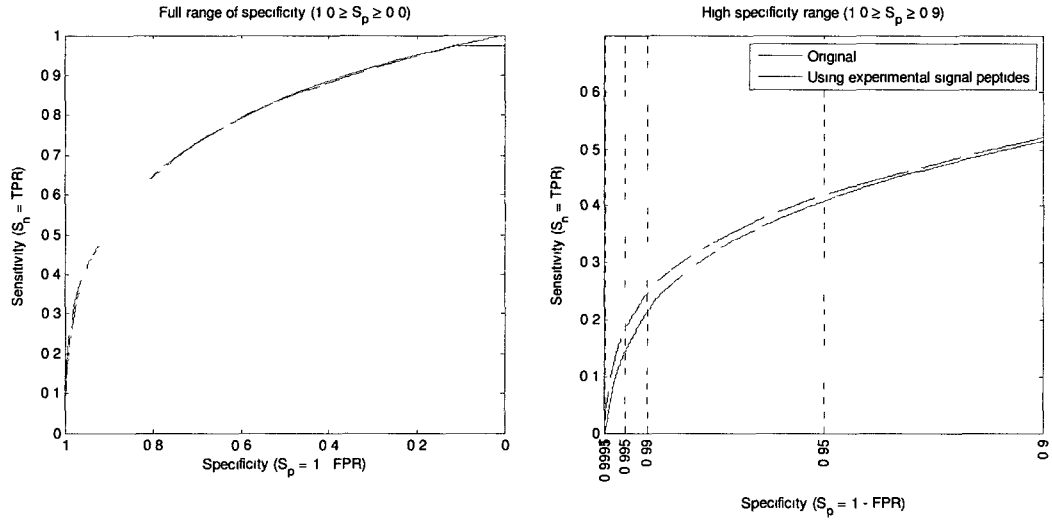


Model using both experimental signal peptide data and experimental transmembrane data

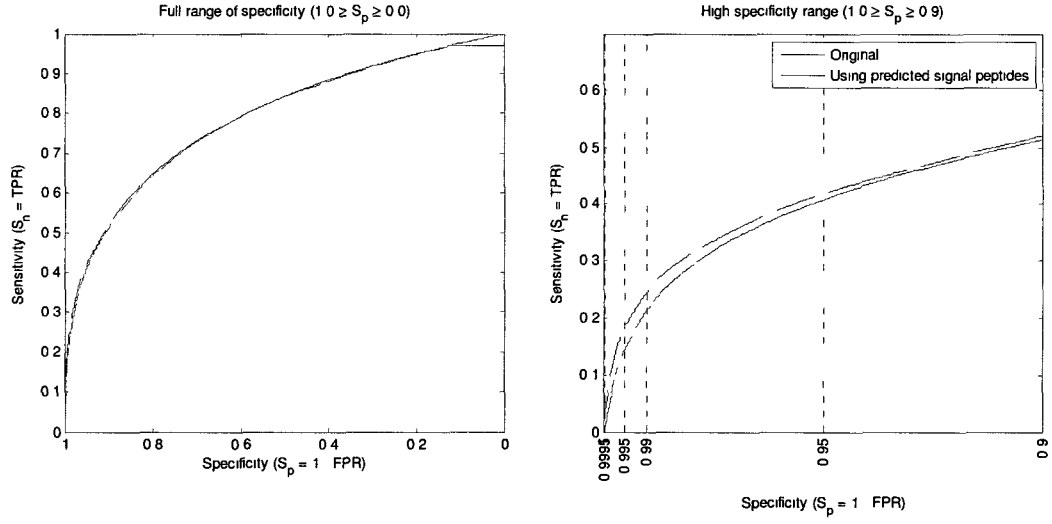


7.2 *H. sapiens*

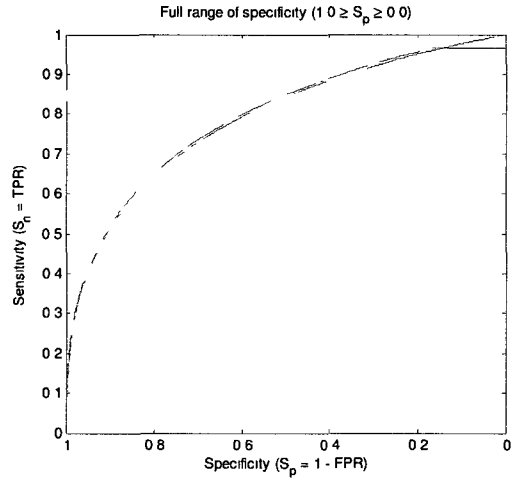
Model using experimental signal peptide data

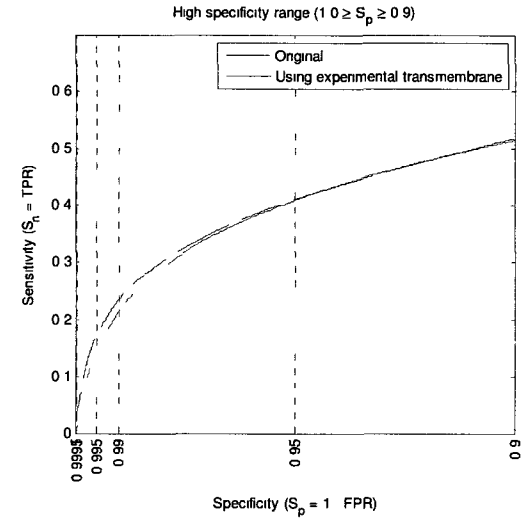


Model using predicted signal peptide data

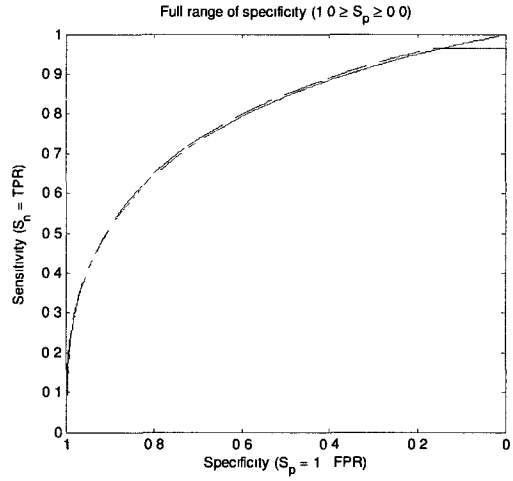


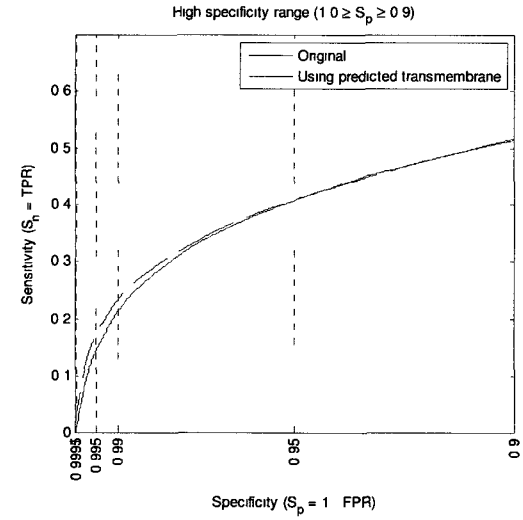
Model using experimental transmembrane region data





Model using predicted transmembrane region data





Model using experimental signal peptide and experimental transmembrane region data