Journal of Bioinformatics and Computational Biology
Vol. 7, No. 6 (2009) 973–990
© Imperial College Press

Imperial College Press
www.icpress.co.uk

SIRIUS PSB: A GENERIC SYSTEM FOR ANALYSIS OF BIOLOGICAL SEQUENCES

CHUAN HOCK KOH[*,†,§], SHARENE LIN[*,¶], GREGORY JEDD[‡,‖]
and LIMSOON WONG[*,**]

[*]*School of Computing, National University of Singapore*
COM1, Computing Drive, 117417, Singapore

[†]*NUS Graduate School for*
Integrative Sciences and Engineering, 117597, Singapore

[‡]*Temasek Life Sciences Laboratory and Department of Biological Sciences*
National University of Singapore, 117604, Singapore

[§]*kohchuanhock@nus.edu.sg*
[¶]*sharene@alumni.nus.edu.sg*
[‖]*gregory@tll.org.sg*
[**]*wongls@comp.nus.edu.sg*

Received 5 June 2009
Revised 15 August 2009
Accepted 15 August 2009

Computational tools are essential components of modern biological research. For example, BLAST searches can be used to identify related proteins based on sequence homology, or when a new genome is sequenced, prediction models can be used to annotate functional sites such as transcription start sites, translation initiation sites and polyadenylation sites and to predict protein localization. Here we present Sirius Prediction Systems Builder (PSB), a new computational tool for sequence analysis, classification and searching. Sirius PSB has four main operations: (1) Building a classifier, (2) Deploying a classifier, (3) Search for proteins similar to query proteins, (4) Preliminary and post-prediction analysis. Sirius PSB supports all these operations via a simple and interactive graphical user interface. Besides being a convenient tool, Sirius PSB has also introduced two novelties in sequence analysis. Firstly, genetic algorithm is used to identify interesting features in the feature space. Secondly, instead of the conventional method of searching for similar proteins via sequence similarity, we introduced searching via features' similarity. To demonstrate the capabilities of Sirius PSB, we have built two prediction models — one for the recognition of Arabidopsis polyadenylation sites and another for the subcellular localization of proteins. Both systems are competitive against current state-of-the-art models based on evaluation of public datasets. More notably, the time and effort required to build each model is greatly reduced with the assistance of Sirius PSB. Furthermore, we show that under certain conditions when BLAST is unable to find related proteins, Sirius PSB can identify functionally related proteins based on their biophysical similarities. Sirius PSB and its related supplements are available at: http://compbio.ddns.comp.nus.edu.sg/∼sirius

Keywords: Sequence analysis; subcellular localization prediction; polyadenylation site recognition; reticulon search.

1. Introduction

With the advancement of sequencing technologies and mass spectrometry, large amounts of data in the form of genome sequences and protein sequences are produced routinely.[1–3] With so many bio-sequences widely and easily accessible, the next challenge is to mine information from them, and computational prediction models are often used for these tasks. In particular, to be able to determine from a DNA sequence its functional sites such as transcription start sites, translation initiation sites, and polyadenylation sites, has always been of interest to biologists as functional sites influence virtually all aspects of the gene expression process. As for protein sequences, the ability to determine the subcellular localization of the protein from a protein sequence can give biologists clues to the functions of that protein.

There exist numerous approaches in building computer models to carry out biosequence analysis. Here, we focus on one particular approach that has been successfully deployed in many high-quality computer models.[4–8] This approach consists of the following sequential steps: (1) feature generation, (2) feature selection, (3) feature integration, and (4) cascade classifier.

One popular machine learning package that is often used to carry out the feature selection and feature integration step is Waikato Environment for Knowledge Analysis (WEKA).[9] The biggest strength of WEKA is that it has implemented a wide variety of feature selection and machine learning algorithms. However, being a general machine learning package, it does not support the feature generation and cascade classifier step, which is often used in analyzing biological sequences. As such, WEKA is less comprehensive when analyzing biological sequences.

BLAST[10] is another tool commonly used by biologists when they want to search for sequences in protein databases similar to the sequence they input. It is commonly accepted that sequence homology is indicative of similar structure and function. However, proteins with similar function do not necessarily share sequence homology. For example, a group of proteins that is known to shape the tubular endoplasmic reticulum do not all share similar sequences,[11] but rather share unusually long transmembrane domains that form “hairpins” in the membrane bilayer. The Sirius Prediction System Builder provides a tool for identifying related proteins based on biophysical character.

Sirius Prediction System Builder (Sirius PSB) is a software package designed to enable biologists with little or no computing knowledge to carry out sequence analysis using computational methods with ease and speed via a graphical user interface. They include building high-quality prediction models, visualization, multi-dimensional protein search and more.

2. Results

In this paper, we used three examples to illustrate the capabilities and potential of Sirius PSB. To prove that Sirius PSB is indeed capable of producing decent prediction and search for real-life applications, we have built two prediction models using Sirius PSB and compared them against current state-of-the-art models. One model is built to predict the subcellular localization of proteins while the other model is designed to carry out recognition of Arabidopsis polyadenylation sites. We also demonstrate how Sirius PSB can make up for certain shortcomings in BLAST via the use of a group of proteins that are known to shape the tubular endoplasmic reticulum.

2.1. *Subcellular localization of proteins*

Given a protein sequence, it is of interest to know the subcellular localization of the protein because it helps us to better understand its functions. Many prediction models have been constructed previously to predict a protein's subcellular localization based on its sequence — in particular, TargetP[12,13] is one such model. It has achieved a high sensitivity (> 85%) and is still often used by biologists today. Hence, we will use TargetP as a means of comparison against our protein localization model. We call the model we generated Protein Localization model (PL model).

The results for TargetP were extracted from Emanuelsson *et al.* (2000).[13] As the authors of TargetP used 5-fold cross-validation, we also ran 5-fold cross-validation on PL model in order to compare with TargetP on equal grounds (Table 1).

The dataset used here was downloaded from the TargetP website. All sequences were extracted from Swiss-Prot and are redundancy reduced. Please refer to Ref. 13 for more details on the preparation of the dataset. The dataset has two versions, plant and non-plant. For the plant version, it contains 141 cTP, 368 mTP, 269 SP and 162 "others" sequences. For the non-plant version, it contains 371 mTP, 715 SP and 1652 "others" sequences. The abbreviations used subsequently are as

Table 1. Prediction performance based on 5-fold cross-validation of TargetP and PL model. Plant sensitivity, Non-plant sensitivity and Overall sensitivity based on equal weightage of each category (based on absolute numbers of each category). As usual, TP refers to the number of true-positive predictions, FN the number of false-positive predictions, and SN the sensitivity level.

			TargetP			PL model		
Set	Category	Size	TP	FN	SN	TP	FN	SN
Plant	cTP	141	120	21	0.851	127	14	0.901
	mTP	368	300	68	0.815	302	66	0.821
	SP	269	245	15	0.911	247	22	0.918
	Others	162	137	25	0.846	151	11	0.932
Plant Sensitivity			0.856 (0.853)			**0.893 (0.880)**		
Non-plant	mTP	371	330	41	0.889	337	34	0.908
	SP	715	683	32	0.955	623	92	0.871
	Others	1652	1451	201	0.878	1610	42	0.975
Non-plant Sensitivity			0.907 (0.900)			**0.918 (0.939)**		
Overall Sensitivity			0.878 (0.888)			**0.906 (0.924)**		

follows: cTP stands for chloroplast transit peptides, mTP stands for mitochondrial targeting peptides, SP stands for signal peptides and "others" stands for peptides in other localizations apart from those mentioned above.

2.2. *Recognition of polyadenylation sites from Arabidopsis genomic sequences*

Polyadenylation is a post-transcriptional process, which cleaves and adds approximately 200–300 adenosine residues to the pre-mRNA 3′ end. This process is an essential processing event and an integral part of gene expression.[14] Having the ability to accurately predict polyadenylation sites allows us to define gene boundaries, predict the number of genes as well as better understand the process.

Currently, the best prediction model for recognition of polyadenylation site for Arabidopsis sequences is designed by us [Koh *et al.* (2007) model].[5] Here, we show that with Sirius PSB, we can build a better model in considerably less time and fewer steps. We call the new model Arabidopsis Polyadenylation Site model (APS model) (Table 2).

The datasets used here were provided by Qingshun Quinn Li.[15] For any two sequences with more than 70% similarity using pair-wise global alignment, one is removed. After redundancy is reduced, the dataset contains 6209 sequences with EST-supported polyadenylation sites, 1501 coding region sequences, 864 5′UTR region sequences and 1581 intronic region sequences. Each sequence is of length 400 and for the EST-supported sequences, the polyadenylation site is at position 301. The dataset split and used is similar to that in Ref. 5 in order to have better comparison. Please refer to Ref. 5 for more details on the preparation of the dataset.

The performance measure used is equal-error-rate value (i.e. the points where sensitivity = specificity).

$$\text{Sensitivity(SN)} = \text{TP}/(\text{TP} + \text{FN}), \tag{1}$$

$$\text{Specificity(SP)} = \text{TN}/(\text{TN} + \text{FP}), \tag{2}$$

Table 2. Equal-error-rate of Koh *et al.* (2007) model and APS model.

Control sequences		Koh *et al.* (2007) model[5] sensitivity & specificity	APS model sensitivity & specificity
5′UTR	SN_0	0.849	**0.871**
	SN_10	0.892	**0.917**
	SN_30	0.915	**0.940**
Coding	SN_0	0.943	**0.967**
	SN_10	0.965	**0.977**
	SN_30	0.975	**0.984**
Intronic	SN_0	0.711	**0.755**
	SN_10	0.788	**0.871**
	SN_30	0.830	**0.921**

where TP (True Positive) is the total number of EST-supported polyadenylation sites that are correctly predicted. FN (False Negative) is the total number of EST-supported polyadenylation sites that are not identified. TN (True Negative) is the total number of sites with prediction score $\leq$ threshold in the ($-$ve) sequences. FP (False Positive) is the total number of sites with score $>$ threshold in the ($-$ve) sequences. SN_0 means that the predicted polyadenylation site is exactly the same as the EST-supported polyadenylation site. SN_10 means the EST-supported polyadenylation site is within 10 nt of the predicted polyadenylation site. SN_30 means the EST-supported polyadenylation site is within 30 nt of the predicted polyadenylation site.

2.3. *Shaping the tubular endoplasmic reticulum (ER) in Saccharomyces cerevisiae*

Voeltz *et al.*[11] have shown that a group of three proteins, RTN1, RTN2 and YOP1 are critical in stabilizing the ER membrane tubules. These three proteins have similar functions, but YOP1 does not share primary sequence homology with RTN1 and RTN2. That is, if RTN1 or RTN2 were input as a query into BLAST, YOP1 would not show up as a hit. However, by using the Nearest Neighbor Search (NNSearch) application in Sirius PSB, with RTN1 and RTN2 as query and setting some constraints (Table 3), YOP1 is returned within the top five hits (Table 4).

Voeltz *et al.*[11] initially only knew that RTN1 and RTN2 were involved and discovered YOP1 through time-consuming biochemical experiments. This could have been avoided or reduced to a large extent if Sirius PSB was used. Furthermore, Voeltz *et al.*[11] believed that there are more proteins needed in stabilizing membrane tubules especially during stress conditions. In this instance, the top hits returned by Sirius PSB could be used as potential candidates.

NNSearch finds YOP1 because it does not simply look for sequence similarities. Instead, it carries out the search based on features like amino acid composition and physiochemical properties such as hydrophobicity, charge, and mass. Looking at the hydrophobicity of the three proteins using Sequence Visualizer provided by Sirius PSB (Fig. 1), it is obvious that all three proteins share two unusually long hydrophobic regions. It is believed that these two hydrophobic regions are critical in stabilizing the tubular ER membrane.[11] Another interesting property of RTN1, RTN2 and YOP1 can be discovered by looking at the hydrophilicity plot using the Sequence Visualizer in Sirius PSB (Fig. 2). In this plot, it can be observed that these proteins have three hydrophilic regions with both of the protein ends being

Table 3. Constraints identified using Sequence Visualizer on RTN1 and RTN2.

No.	Constraints
1	Having three or more hydrophilic regions with value $>$ 10 and size $>$ 10
2	Most hydrophobic region has value $>$ 90

Table 4. Top 10 hits of NNSearch with RTN1 & RTN2 as query and constraints listed in Table 3.

No.	Sequence header
1	sp|Q04947|RTN1_YEAST Reticulon-like protein 1 OS=Saccharomyces cerevisiae GN=RTN1
2	sp|Q12443|RTN2_YEAST Reticulon-like protein 2 OS=Saccharomyces cerevisiae GN=RTN2
3	sp|P23641|MPCP_YEAST Mitochondrial phosphate carrier protein OS=Saccharomyces cerevisiae GN=MIR1
4	sp|Q12402|YOP1_YEAST Protein YOP1 OS=Saccharomyces cerevisiae GN=YOP1
5	sp|P53633|PRA1_YEAST Prenylated Rab acceptor 1 OS=Saccharomyces cerevisiae GN=YIP3
6	sp|P00410|COX2_YEAST Cytochrome c oxidase subunit 2 OS=Saccharomyces cerevisiae GN=COX2
7	sp|P39692|MET10_YEAST Sulfite reductase [NADPH] flavoprotein component OS=Saccharomyces cerevisiae GN = MET10
8	sp|Q06142|IMB1_YEAST Importin subunit beta-1 OS=Saccharomyces cerevisiae GN=KAP95
9	sp|P40069|IMB4_YEAST Importin subunit beta-4 OS=Saccharomyces cerevisiae GN=KAP123
10	sp|P38329|YB85_YEAST Uncharacterized membrane protein YBR235W OS=Saccharomyces cerevisiae GN=YBR235W

hydrophilic. Also, it can be seen that one end of the hydrophilic region is relatively longer than the other end.

With these two plots (Figs. 1 and 2), one can easily hypothesize how these proteins might interact with the membrane (Fig. 3). This is in good agreement with experimental evidence of the topologies of reticulon proteins.[16]

3. Discussion

From Table 1, it is clear that using the PL model generated by Sirius PSB produces better results in the prediction of subcellular localization of proteins compared to TargetP. Although the results of the PL model are only 1–4% higher than TargetP, it should be noted that the PL model is built using Sirius PSB effortlessly in a greatly minimized time span. This was accomplished simply by using features found in Sirius PSB such as 1-, 2- and 3-gram (A k-gram feature is simply a string of k consecutive characters).

From Table 2, APS model has shown improved performance over our previous model.[5] What we would like to stress here is that even though both methods followed the same computational approach, our previous model[5] was designed by writing several programs and having to change the codes in the programs whenever we wanted to try out different settings or deploy different features. In contrast, Sirius PSB encompasses settings that can be changed instantly via a few mouse clicks.

Another important difference is that the 261 candidate features used for our previous model[5] were decided upon after spending a lot of time and effort searching and reading literature about the Arabidopsis polyadenylation process. Compare

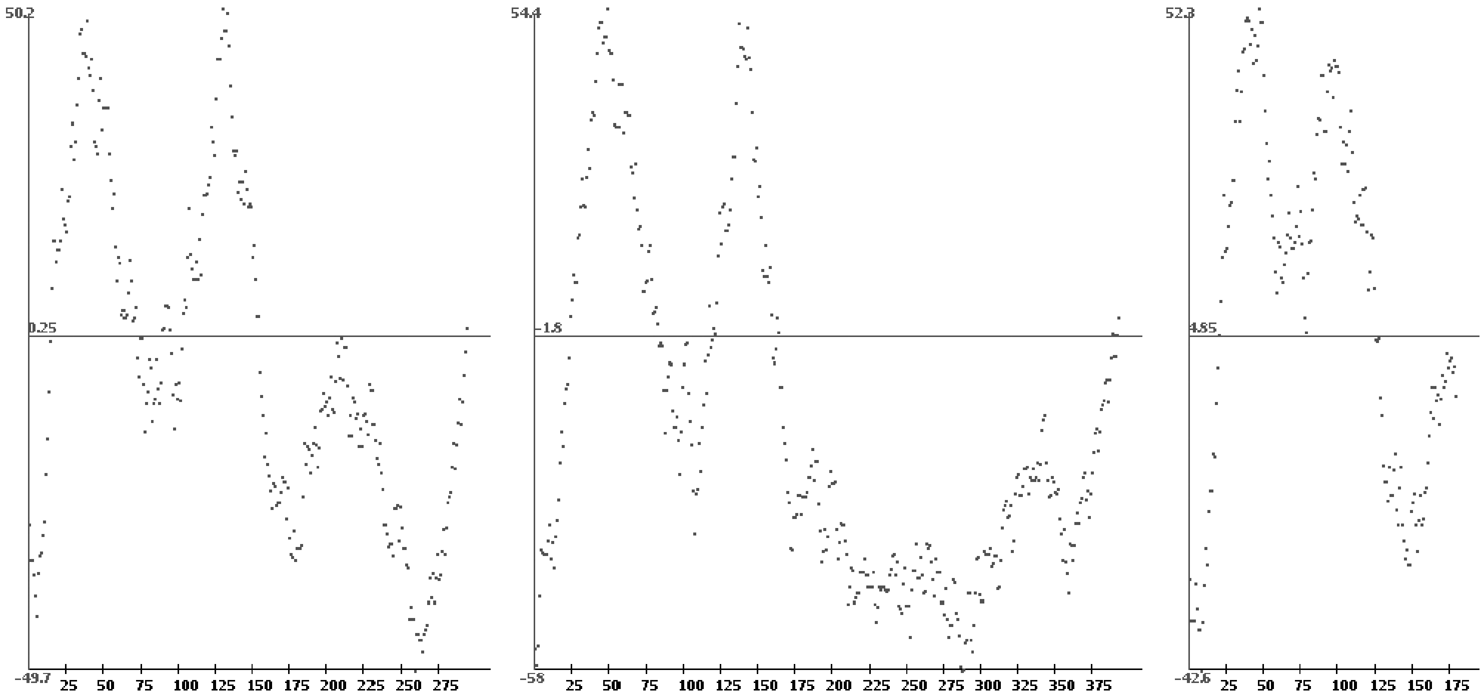


Fig. 1. Graphs of RTN1, RTN2 and YOP1 respectively drawn using Sequence Visualizer where a signature “M” can be seen in all three graphs which is due to two unusually long hydrophobic regions. (y-axis: Hydrophobic Index, x-axis: Residue Number, Settings: WindowSize = 30).

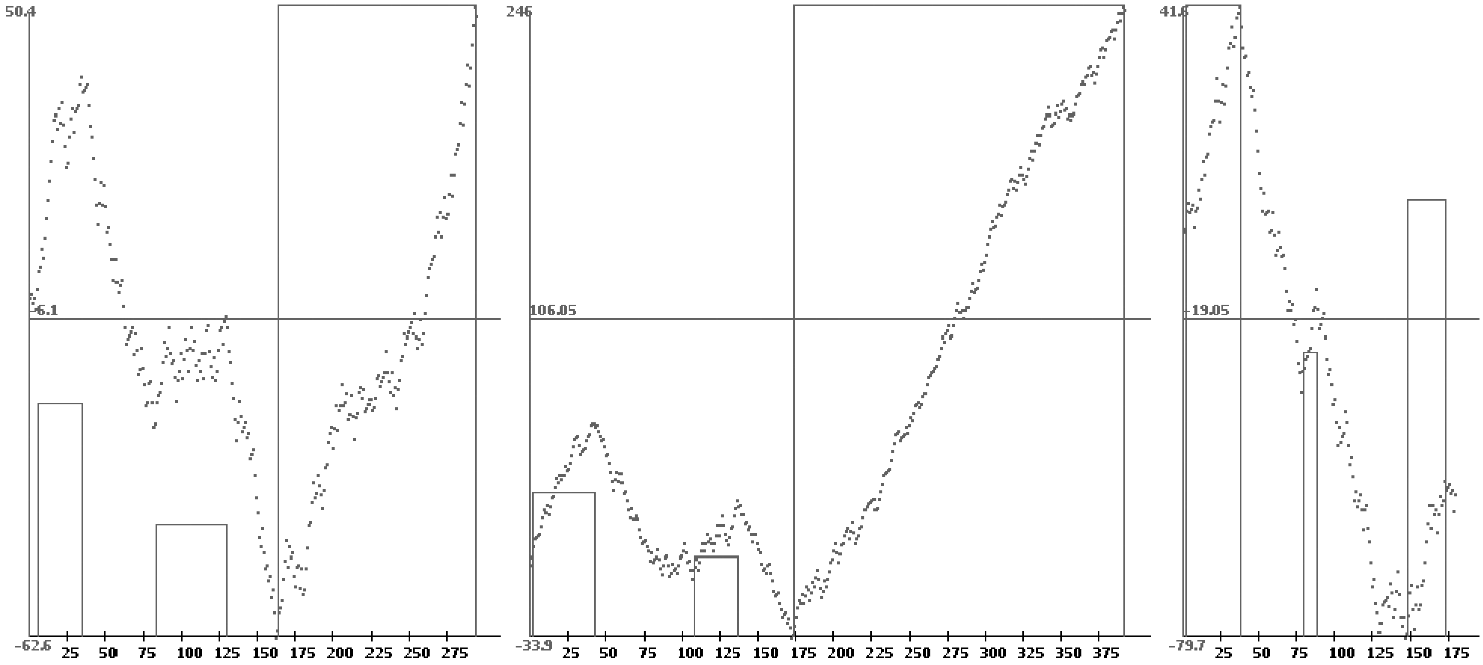


Fig. 2. Graphs of RTN1, RTN2 and YOP1 respectively drawn using Sequence Visualizer where each has three hydrophilic regions with one end being notably longer than the other. (y-axis: Hydophilic Index, x-axis: Residue Number, Settings: ValueCutOff = 15).

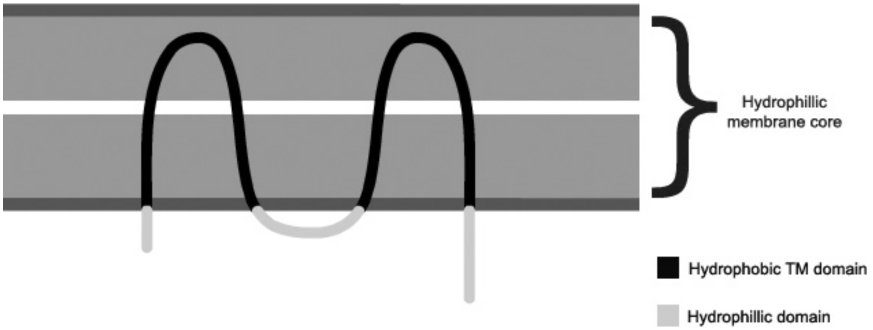


Fig. 3. Possible representation of how RTN1, RTN2 and YOP1 would interact with the membrane. Two unusually long hydrophobic regions slot in between the three hydrophilic regions with one end relatively longer than the other.

this with the APS model which auto-generated 144 features simply by running the genetic algorithm (provided by Sirius PSB) on the training dataset. Furthermore, since the only difference between the two models is the set of candidate features used, we can conclude that the 144 auto-generated features are more concise and meaningful biologically than the 261 candidate features. Looking carefully at the set of 144 features, we notice that it does reflect some of the knowledge found in the literature. Significantly, no prior knowledge was fed into the running of the Sirius genetic algorithm. Due to these differences, our previous model[5] took us about several months to complete whereas APS model took us only a few days to build from scratch.

Usually, in order to build a good classifier, users need to have a good understanding of the problem to select a "good" set of features. The automated feature generation of Sirius PSB provides two advantages. Firstly, users no longer need to provide the set of features. Secondly, the auto-generated set of features could in fact give us some knowledge about the problem. As demonstrated, Sirius PSB prediction models not only outperformed current state-of-the-art models in terms of accuracy, but the time and effort required to build them is also significantly reduced. With Sirius PSB, building of classifiers and bio-sequence analysis would be a less tedious process.

Yet another powerful ability of Sirius PSB is the ability to search for similar proteins using user-specified features where BLAST fails. Biologists usually have some inkling as to the important features to look for, and this proves to be very useful in the search. For instance, in the abovementioned example, Voeltz *et al.*[11] know that hydrophobicity is certainly an important component in the proteins. They can then simply input and search for proteins that "look alike" to the query proteins in terms of their hydrophobicity signatures. Even in the case where biologists do not have much information about the query, they can still use visualizing tools provided by Sirius PSB to decide on the type of features that are likely to be important. Although the relationship between RTN1, RTN2 and YOP1 has been previously published, our paper is the first to show that these proteins can be identified as relatives based purely on their biophysical properties. We believe that this provides a clear example of the utility of Sirius PSB.

4. Methods

Sirius PSB has four main operations (applications): (1) Building a classifier (Trainer), (2) Deploying a classifier (Predictor), (3) Searching for proteins similar to query proteins (NNSearcher), and (4) Preliminary and post-prediction analysis (Misc). Sirius PSB supports both protein and DNA sequences. The current version of Sirius PSB is 2.33. Sirius PSB is written in Java and hence should have no problems running on any platform with JVM. To obtain the results mentioned in this paper, users can simply follow the user guide given in the link below. The program (Sirius PSB), datasets used and user guide can be downloaded from Sirius PSB website: (http://compbio.ddns.comp.nus.edu.sg/∼sirius/). The two prediction models (subcellular localization and polyadenylation sites) are also available for use on the website.

4.1. *Subcellular localization of proteins*

For our protein localization model (PL model), we employed the first three steps of the approach — feature generation, feature selection, and feature integration.

For the feature generation step, we used straightforward features of 1-, 2- and 3-gram with window (0,100). Note that there are 20 different amino acids. This means there are $20 + 20^2 + 20^3 = 8420$ features. We then calculated the occurrence of the 8420 features for the first 101 characters of each sequence. In the feature selection step, we filtered away those with chi-square value ≤ 0.

During the feature integration step, we used Support Vector Machine[17] with polynomial kernel of degree two and buildLogisticModels (set to True) for all the sequences except for non-plant SP presequence, where Naive Bayes algorithm was used instead. Like TargetP, we have seven different classifiers for each type of presequence. All of them used SVM except for the non-plant SP preseqeuence. Naive Bayes was used for non-plant SP presequence because poor performance was observed when SVM was used for this particular presequence. This shows the flexibility of Sirius PSB in that the choice of the machine learning method can be changed easily without fuss.

4.2. *Recognition of polyadenylation sites from Arabidopsis genomic*

Feature generation, feature selection, feature integration and cascade classification is the methodology used by both our previous model[5] and APS model. The settings for feature selection (chi-square with threshold 0), feature integration [support vector machine (SMO) with buildLogisticsModel set to True] and cascade classification

$(-40, 41)$ for both models are the same. The only difference between the two models is in the feature generation step.

Our previous model[5] generated 261 candidate features based on biological knowledge from literature. APS model used 144 candidate features auto-generated by running genetic algorithm on training Dataset A.

4.2.1. *Genetic algorithm*

In most classification problems, having a “correct” set of features would ensure high accuracy. However, finding such a set of features can be a daunting task. In Sirius PSB, we have incorporated a genetic algorithm approach to help users find such features. We have also demonstrated the usefulness of this approach by using the polyadenylation example.

To be more specific, the algorithm employed here is a slight variant of the original genetic algorithm. Unlike the original genetic algorithm where each population contains a set of candidate solutions, in our approach, each population is a single solution where each member in the population is a feature.

There are typically two properties to be defined before the running of genetic algorithm can commence — genetic representation and fitness function. Variable length value encoding is chosen to be the genetic representation. Such a representation will cause implementation for crossovers to be more complex, but this is a fair trade-off when taking the diversity of feature types involved into consideration.

As for the fitness function, chi-square was chosen for two main reasons. Firstly, it is efficient to compute. Being computationally efficient is important as it would make the algorithm impractical otherwise. The other reason is that chi-square holds a property whereby in the case that chi-square value of feature A is greater than chi-square value of feature B, we can be sure that feature A is “better” than feature B with respect to the training dataset. This is an important property that must hold true for fitness functions.

Finally, as the diversity of the population is another important factor for the success of genetic algorithm, measures are taken to prevent the convergence of the population. High default mutation rate is set, at 70% (editable by users). Also, an additional step has been added — the elimination of similar features. In this step, if two features are too similar ($> 80\%$), one would be discarded. New randomly generated features will replenish the discarded features. With these strategies deployed, the likelihood of the convergence of the population is significantly reduced.

Detailed description of our genetic algorithm is in Sirius2.33_GeneticAlgorithm. txt which is available from Sirius website. A brief outline of the algorithm is as follows:

Initial population

- Will be randomly generated unless user provides (thus possible to continue from previous runs)

Evaluation

- Fitness score used is the chi-square value
- Chi-square value is calculated based on the dataset used for GA

Selection

- Features are selected with a probability directly proportionate to its chi-square value

Crossover

- Again, features with the higher score have a higher probability to be selected as a parent
- Two offspring will be generated by taking parts from each parent

Mutation

- Features will be randomly selected to undergo mutation

Eliminate similar features

- Eliminate features that are highly similar (>80%) within the population.
- This is to prevent the population from being dominated by similar features

Replenish

- Generate new features randomly to replenish the population due to elimination in the "Eliminate similar features" step

Return to the "Evaluation" step unless termination generation is reached

- It is possible to terminate prematurely if the user requests so

4.3. *Shaping the tubular endoplasmic reticulum in Saccharomyces cerevisiae*

In this instance, we used NNSearcher with RTN1 and RTN2 as the input query, and searched against the Swiss-Prot (*Saccharomyces cerevisiae*) database. For the features, we used 1-gram, 2-gram and a series of physiochemical properties. We also listed down a few constraints derived from Voeltz *et al.*[11] paper and as well as information from the visualizer (provided by Sirius PSB) through the study of RTN1 and RTN2.

4.3.1. *NNSearcher*

NNSearcher is a tool provided in Sirius PSB meant to carry out the searching of similar proteins based on their features. The standard approach in searching for similar proteins would be via BLAST. However, this approach may not always work as it is known that some proteins which are related functionally may not share sequence similarities. This is where NNSearcher could help in filling up the gap. NNSearcher attempts to find features that are unique and well conserved in the query proteins. Then using these features, search for other proteins in the database that are similar to the query proteins.

The algorithm employed by NNSearcher is as follows:

Generate

- Features to generate can be decided by the user
- Default is 1-gram, 2-gram and a series of physiochemical properties

Normalize

- Features' values are normalized to [0,1]

Filter

- $queryDBDifference_i = (queryMean_i - DBMean_i)/DBStdDev_i$
- Filter away feature$_i$ with queryDBDifference$_i$ < 0.5
- Filter away feature$_i$ with queryStdDev$_i$ > 0.5

Assign

- $weight_i = 0.5 * (1/queryStdDev_i) + 0.5 * (queryDBDifference_i)$

Score

- $Score_j = 1 - \Sigma_i |(featureValue_{i,j} - queryMean_i) * weight_i|$
- Calculate the distance of a particular protein from query proteins

Rank

- Sort by descending order of score
- Higher score would mean greater similarity to query proteins

Notations:

- $queryMean_i$ – Mean value of query for feature$_i$
- $queryStdDev_i$ – Standard deviation of query for feature$_i$
- $DBMean_i$ – Mean value of database for feature$_i$
- $DBStdDev_i$ – Standard deviation of database for feature$_i$
- $featureValue_{i,j}$ – Value of protein$_j$ for feature$_i$

In the filtering step, NNSearcher filters away features where the query is similar to the majority in the database (queryDBDifference$_i$ < 0.5) because this would mean that those features are not unique to the query proteins. Also, it filters away features that are not well conserved (queryStdDev$_i$ > 0.5) in the query proteins.

In assigning weights to features, there are two factors to consider. Firstly, features that are well conserved (1/queryStdDev$_i$) in the query proteins are given heavier weights. Also, heavier weights are given to features that are unique (queryDBDifference$_i$) to the query proteins in comparison to the database proteins.

In the scoring step, NNSearcher attempts to find proteins that are similar to the query proteins based on those features identified to be unique and well conserved to the query proteins ($1 - \Sigma_i |(featureValue_{i,j} - queryMean_i) * weight_i|$).

5. Sirius PSB Overview

As mentioned, Sirius PSB has four main operations facilitated by four applications: (1) Trainer — Enable the building of classifiers, (2) Predictor — Allows the usage of classifiers built using Trainer, (3) NNSearcher — Searching for proteins similar to query proteins based on features' similarity, and (4) Misc — Includes a variety of tools for preliminary and post-prediction such as redundancy reduction and visualization. All these applications are integrated seamlessly into one environment (Sirius PSB) to allow users to carry out sequence analysis with ease and convenience (Fig. 4).

5.1. *Trainer*

For sequence prediction problems, there are generally three different categories. One is where prediction is made only once for each sequence (e.g. subcellular localization of proteins). Another is where there is a known anchor motif and prediction is only made when the anchor motif is encountered (e.g. translation start site — ATG). The last type is where no anchor motif exists and every position of the sequence is a candidate site and prediction has to be made on every position of the sequence.

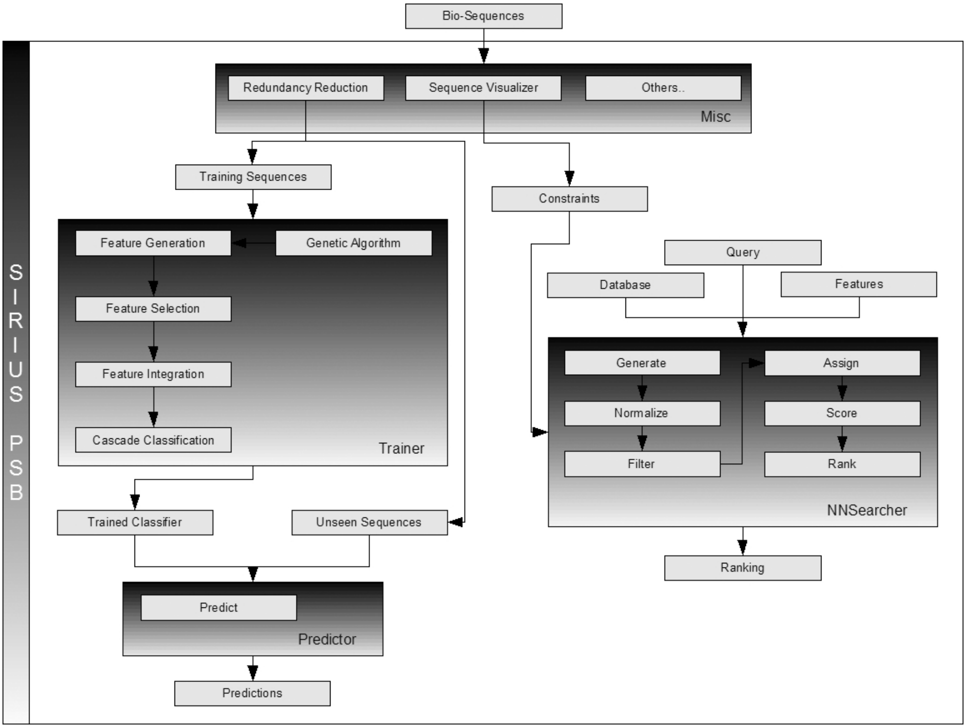


Fig. 4. Sirius PSB software architecture.

Trainer is capable of building classifiers for any of these categories by using a general approach — feature generation, feature selection, feature integration, and cascade classification.[5–8]

Feature generation is the most critical step in this four-step approach. When given a "correct" set of candidate features, one can easily build a high-accuracy classifier from them. However, the task of finding the "correct" set of candidate features is hard. Therefore, Trainer provides an option for the auto-generation of features using Genetic Algorithm. We have shown in this paper that high-quality prediction models can be built with features auto-generated by Genetic Algorithm. The details of Trainer's Genetic Algorithm are stated in Sec. 4.2.1.

As mentioned earlier, WEKA[9] is a popular and constantly updated machine learning package with a wide variety of feature selection and machine learning algorithms. WEKA is also an open-source software issued under the GNU General Public License. The ability to utilize this rich resource would certainly be an advantage. Hence, Trainer accesses WEKA via direct Java function calls whenever feature selection algorithm or machine learning algorithm is needed.

5.2. *Predictor*

The Predictor application allows the use of classifiers built using Trainer to make prediction on bio-sequences that have unknown properties. In addition, it contains user-friendly options like sorting sequences based on prediction scores, graphical display of prediction scores and can also limit the predictions to user-defined motifs if the user has a specific list of motifs that he/she is interested in.

5.3. *NNSearcher*

The NNSearcher application is built specifically for finding proteins that are similar to input proteins based on the similarity of features. Its algorithm is listed under Sec. 4.3.1. Although prior knowledge is not required to run NNSearcher, but if present, more refined searches can be obtained. To this end, users can utilize Sequence Visualizer tool in the Misc application.

5.4. *Misc*

The Misc application encompasses a variety of tools to assist users in pre- and post-bio-sequences analysis. For example, Sequence Visualizer tool is used to obtain understanding of RTN1 and RTN2 which leads to the definition of two constraints (Table 3). Another useful tool is Redundancy Reduction, as this tool allows users to reduce sequence similarity in the dataset to prevent biased validation results.

6. Conclusion

In this paper, we used three real-life problems to demonstrate some of the capabilities of Sirius PSB. We have also shown that Sirius PSB can assist in solving these problems in a seamless and hassle-free manner.

As demonstrated, two prediction models were built using Sirius PSB, and not only did these prediction models outperform current state-of-the-art models in terms of accuracy, but the time and effort required to build them is also significantly reduced. With Sirius PSB, development of high-quality prediction models is no longer limited to skilled programmers.

We have also developed a novel way to perform searches for similar proteins. This new approach of searching for similar proteins based on features is particularly useful when the proteins do not exhibit similarities in their sequences. In situations like this, the standard approach like using BLAST would not yield satisfactory results. This is where Sirius PSB can fill the gap and produce plausible suggestions to the query.

It is not possible to expound the full capabilities of Sirius PSB in this short paper. Readers are encouraged to download Sirius PSB at the given link to better understand Sirius PSB and explore its other abilities — Sirius PSB can run on any operating system with JVM and has a friendly graphical user interface to guide the user along.

Acknowledgments

This work was supported in part by Singapore National Research Foundation grant NRF-G-CRP-2997-04-082(d) (Wong), a National University of Singapore NGS scholarship (Koh) and the Temasek Life Sciences Laboratory and Singapore Millennium Foundation (Jedd).

References

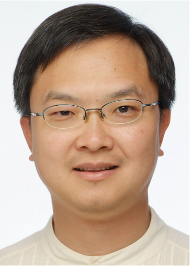
1. Shendure J, Mitra RD, Varma C, Church GM, Advanced sequencing technologies: Methods and goals, *Nat Rev Genet* **5**:335–344, 2004.
2. Aebersold R, Mann M, Mass spectrometry-based proteomics, *Nature* **422**:198–207, 2003.
3. Chi KR, The year of sequencing, *Nat Methods* **5**:11–14, 2008.
4. Liu H, Han H, Li J, Wong L, DNAFSMiner: A web-based software toolbox to recognize two types of functional sites in DNA sequences, *Bioinformatics* **21**:671–673, 2005.
5. Koh CH, Wong L, Recognition of polyadenylation sites from Arabidopsis genomic sequences, in *Proceedings of 18th International Conference on Genome Informatics*, 3–5 December, Singapore, pp. 73–82, 2007. Supplementary information: Koh *et al.*, Model Datasets & Source codes, 2007 [http://www.comp.nus.edu.sg/∼wongls/projects/dnafeatures/giw07-supplement/]
6. Liu H, Han H, Li J, Wong L, An *in-silico* method for prediction of polyadenylation signals in human sequences, in *Proceedings of 14th International Conference on Genome Informatics*, December, Yokohama, pp. 84–93, 2003.
7. Liu H, Han H, Li J, Wong L, Using amino acid patterns to accurately predict translation initiation sites, *In silico Biol* **4**:255–269, 2004.
8. Liu H, Wong L, Data mining tools for biological sequences, *J Bioinform Comput Biol* **1**:139–167, 2003.
9. Witten IH, Frank E, *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd ed., Morgan Kaufmann, San Francisco, 2005.
10. BLAST [http://blast.ncbi.nlm.nih.gov/Blast.cgi]

11. Voeltz GK, Prinz WA, Shibata Y, Rist JM, Rapoport TA, A class of membrane proteins shaping the tubular endoplasmic reticulum, *Cell* **124**:573–586, 2006.
12. TargetP 1.1 Server [http://www.cbs.dtu.dk/services/TargetP/]
13. Emanuelsson O, Nielsen H, Brunak S, Heijne GV, Predicting subcellular localization of proteins based on their N-terminal amino acid sequence, *J Mol Biol* **300**:1005–1016, 2000.
14. Loke CJ, Stahlberg EA, Strenski DG, Haas BJ, Wood PC, Li QQ, Compilation of mRNA polyadenylation signals in Arabidopsis revealed a new signal element and potential secondary structures, *Plant Physiol* **138**:1457–1468, 2005.
15. Ji G, Zheng J, Shen Y, Wu X, Jiang R, Lin Y, Loke JC, Davis KM, Reese GJ, Li QQ, Predictive modeling of plant messenger RNA polyadenylation sites, *BMC Bioinformatics* **8**:43, 2007.
16. Yang YS, Strittmatter SM, The reticulons: A family of proteins with diverse functions, *Genome Biol* **8**:234, 2007.
17. Cortes C, Vapnik V, Support-vector networks, *Machine Learning* **20**:273–297, 1995.

Chuan Hock Koh is a Ph.D. candidate at the National University of Singapore under a scholarship from the National University of Singapore Graduate School for Integrative Sciences and Engineering. He received his B.S. in Computational Biology from the National University of Singapore in 2008. He is currently situated at Human Genome Center, Institute of Medical Science, University of Tokyo, Japan, as a visiting graduate student.

Sharene Lin graduated from the National University of Singapore with a B.S. in Computing in 2007. Her current interest mainly revolves around communicating complex computational ideas effectively to biologists and the general public.

Gregory Jedd received his B.S. from Stanford University, USA, and earned his Ph.D. in Cell Biology from the University of Chicago, USA. He did his postdoctoral work at the Rockefeller University, USA, and since 2004 has been a Principle Investigator at the Temasek Life Sciences Laboratory in Singapore. His group employs a multidisciplinary approach to investigate fundamental questions in cellular assembly.



Limsoon Wong is a Professor of Computer Science and Professor of Pathology at the National University of Singapore. He is currently working mostly on knowledge discovery technologies and is especially interested in their application to biomedicine. Limsoon has written about 150 research papers, a few of which are among the best cited of their respective fields. He serves on the editorial boards of *Information Systems* (Elsevier), *Journal of Bioinformatics and Computational Biology* (ICP), *Bioinformatics* (OUP), and *Drug Discovery Today* (Elsevier). He received his B.Sc. (Eng) in 1988 from Imperial College London, UK, and his Ph.D. in 1994 from University of Pennsylvania, USA.