Predicting Vertebrate Promoters with Homogeneous Cluster Computing

Fang-Yie Leu, Neng-Wen Lo*, Lun-Ni Yang
*Department of Computer Science and Information Engineering and *Department of Animal Science and Biotechnology, Tung-Hai University, Taiwan*
leufy@thu.edu.tw , nlo@thu.edu.tw , g922810@thu.edu.tw

Abstract

This article proposes a system, named Vertebrate Promoter Prediction System (VPPS), which employs a new approach to predict vertebrate promoters using statistical techniques. We analyze a putative promoter sequence by investigating the presence of short promoter-specific sequences and known transcription factor binding sites. A gene-based consensus sequence-extracting program (GCSEP) is developed to extract promoter and non-promoter specific patterns, 6-20 bps in length. Applying a weighed-matrix, we can more easily manipulate the extracted patterns and accurately predict whether and where an unknown DNA sequence contains promoters. Furthermore, cluster computing is deployed to accelerate the weighed-matrix manipulation and K-gram parse. Our experimental results show that the VPPS has better true positive and lower false positive rates than other prediction tools.

1 Introduction

A promoter is a deoxyribonucleic acid (DNA) sequence locating on the upstream of a transcriptional starting site (TSS). Fig.1 shows the location of a promoter and how a simple gene is transcribed from DNA to mRNA. Presently certain intrinsic features embedded in a promoter, e.g., TATA BOX, CCAAT BOX and CpG islands, have been identified.

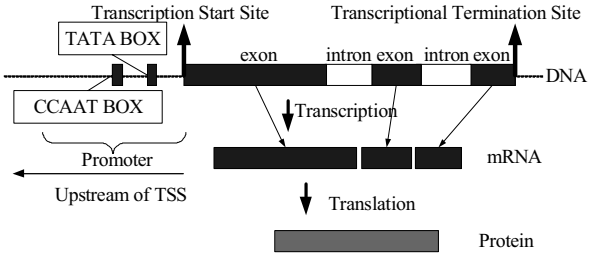

Fig. 1 A schematic diagram representing how a gene is transcribed and translated

Promoter region, or simply promoter, is responsible for initiating gene transcription. As Human Genome Project's (HGP) first phase ends, deciphering the characteristics of a gene promoter has become one of the key research tasks. Once the common hallmarks of promoter sequences are characterized, molecular biologists will be able to gain insights into the function of a gene and realize how the gene is expressed in terms of its specificity and intensity.

In this article, we propose a system, named Vertebrate Promoter Prediction System (VPPS), which employs a statistics-based new approach to predict promoter regions within a given unknown DNA sequence. VPPS first extracts vertebrate promoter and non-promoter sequences from EPD (eukaryote promoter database) [5] and NCBI (National Center for Biotechnology Information) GenBank[8], respectively, and partitions them into fixed length of segments, 6-20 bps, in order to identify the possible transcription factor binding sites (TFBSs), which are sequence segments segmented from promoters, and non-transcription factor binding sites (NTFBSs), which are also segments but segmented from non-promoters, both of which are stored in Total Transcription Factor Database (TTFDB), a database holding total TFBSs and NTFBSs. Non-promoters may be RNAs or genome sequences. TFBSs and NTFBSs are used to rank an unknown sequence S. VPPS finally predicts if S contains promoters or not by comparing it against TTFDB. When a promoter is discovered, like other research projects, we assume that an anticipated gene would likewise exist downstream of the promoter.

2 Related work

Pedersen et al. [4] and Werner [7] provided a detailed survey, from biological viewpoint, on the fundamental structure of a promoter and eukaryotic promoter prediction. Pedersen also brought up the idea that identifying the presence of some characteristics, e.g., TATA box and CpG islands, is helpful in predicting a promoter. Werner thought that both of promoters and non-promoters as well as exon and intron should be analyzed. Fickett et al. [1] and Ohler

et al. [3] reviewed promoter prediction researches, all from biological and computational prediction viewpoints.

3 Architecture of VPPS

VPPS consists of two subsystems, Transcription Factor Finder (TFF) and Promoter Positioner (PP) as shown in Fig. 2. The former is responsible for generating TFBSs and NTFBSs by invoking our developed Gene-based consensus sequence-extracting program (GCSEP). The latter predicts if a sequence is a promoter or non-promoter.

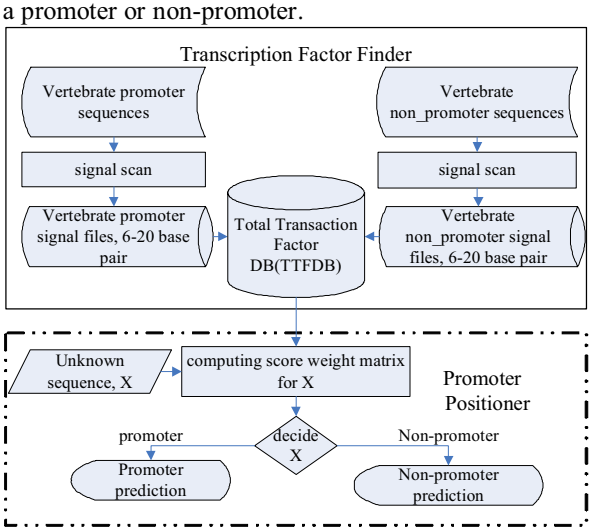

Fig. 2 System architecture of VPPS and its processing flow

TFBSs, the basic units extracted from promoters, are formally defined as the minimal complement of proteins necessary to reconstitute accurate transcription from a minimal promoter (such as a TATA element or initiator sequence)[9].

Like N-gram, a promoter or non-promoter with n bps in length in this research is segmented along its sequence into n-K+1 K-bps patterns, named K-grams, K=6,7,8,…,20. Those segmented from promoters are TFBSs, and those from non-promoters are NTFBSs . A TFBS (NTFBS) type K-gram is scored as it is generated from training promoters (non-promoters). However, the score of a NTFBS type K-gram is negative. Let P = $\{TFBS_1, TFBS_2,\ldots,TFBS_q\}$, P's scores are Sp = $\{SC_1, SC_2, \ldots, SC_q\}$, $SC_i > 0$, i=1,2,…,q. N = $\{NTFBS_1, NTFBS_2,\ldots,NTFBS_m\}$, N's scores are SN = $\{SCN_1, SCN_2, \ldots, SCN_m\}$, $SCN_j < 0$, j = 1, 2,…,m. If a pattern, say PT, appears in both P and N, say $TFBS_i$ and $NTFBS_j$ respectively, then $SC_i = SC_i + SCN_j$ and N = N-$\{NTFBS_j\}$. While a pattern appears either in P or in N but not both, its score does not change. We merge P and N and sort the result based on lexicographic order. Finally, all PTs of which score=0 are deleted and the remainders are stored in TTFDB[6].

Actually, TTFDB's pattern scores can be generated by three ways.

(1). For any K, K=6,7,8…,20, we can generate 4^K patterns, each sequentially compares with K-grams which are dynamically extracted from training promoters and non-promoters. If there are a total of T training sequences and each is L in length, the segmentation effort (times) is

$$\sum_{K=6}^{20}(4^K * T * (L-K+1)) \ldots\ldots(1)$$

The search effort is

$$\sum_{K=6}^{20} 4^K (\log_2 |TFDB|) \ldots\ldots(2)$$

(2). We segment each training sequences of length L into L-K+1 patterns, each compares with 4K patterns previously generated by means of binary search. The segmentation effort is

$$\sum_{K=6}^{20} T * (L-K+1) \ldots\ldots(3)$$

The search effort is

$$\sum_{K=6}^{20} T * (L-K+1) * 2K \ldots\ldots(4)$$

(3). VPPS segments each training sequence of length L into L-K+1 patterns, and accumulates the times a pattern is generated. The segmentation effort is

$$\sum_{K=6}^{20} T * (L-K+1) \ldots\ldots(5)$$

The search effort is

$$\sum_{K=6}^{20} \log_2 (T * (L-K+1)/2) \ldots\ldots(6)$$

3.1 Transcription Factor Finder (TFF)

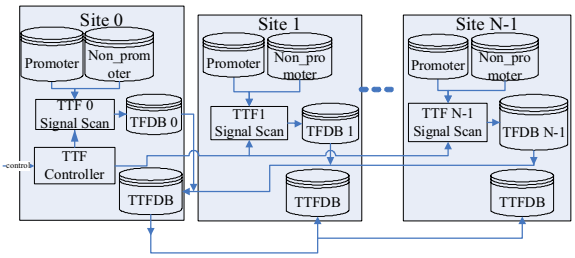

Fig. 3 Transcription Factor Finder architecture

TFF employs a cluster system of N sites shown in Fig.3 to speedup data processing. The tasks of generating TFBSs and NTFBSs are distributed to

cluster nodes, therefore, each has its locally generated result. All the results should come together before PP can position possible promoters. Algorithm 1 shows the control function deployed to estimate distribution and balance processing loads. GCSEP is implemented by algorithm 2 and 3.

Algorithm1: TFF∅ Controller /*Performed by site0.

Input: Number of training sequences, say M=|P|+|NP|, and number of cluster sites, say N, where P and NP are sets of promoters and non-promoters.

Output: Training sequences distributed to cluster sites.

Method: Based on site$_i$'s performance Pc_i [2] to allocate sequences TS_i to site$_i$, $|TS_i|=\left\lceil \frac{Pc_i}{\sum_{j=0}^{N-1} Pc_j} * M \right\rceil$, i=0,1,2,…N-1……(7)

Algorithm2: TFFi Signal Scan /*performed by site$_i$, i=0,1,2,…,N-1.

Input: TS_i, consisting of P_i and NP_i.

Output: $TFDB_i$ and score$_i$[].

Method:

1. For each promoter $\in P_i$, say P_s of length L_s
   Partition P_s into (L_s-k+1) K-grams, k=6,7,…,20; save them in Pk.
   /*Assuming there are a total of q K-grams, $q \le \sum_{\forall Ps} \sum_{k=6}^{20} (L_s - k + 1)$
2. Calculate the scores for the q K-grams to obtain score$_i$[].
3. For each non-promoter $\in NP_i$, say P_t of length L_t
   Partition P_t into (L_t-k+1) K-grams, k=6,7,…,20; save them in NPk.
   /*Assuming there are a total of r K-grams, $r \le \sum_{\forall Pt} \sum_{k=6}^{20} (L_t - k + 1)$
4. Calculate the scores for the r K-grams to obtain score1$_i$[].
5. For each K-gram, say KG, appearing both in Pk and NPk
   {score$_i$[KG]=score$_i$[KG]+score1$_i$[KG] ; delete KG from NP} /*score1$_i$[KG]<0
6. Merge (Pk, scpre$_i$[]) and (NPk, score1$_i$[]); sort the result based on lexicographic order; save the sorted result in $TFDB_i$.
7. Except site$_0$, send ($TFDB_i$, score$_i$[]) to site$_0$.

Algorithm3: Merge all $TFDB_i$ into TTFDB. /*performed by site$_0$

Input: $TFDB_i$, i=0,1,2,…N-1.

Output: TTFDB and the score table for K-grams in TTFDB, say scoreT[].

Method:

1. Merge all TFDBi to form TTFDB by using multiway merging algorithm and $scoreT[X] = \sum_{i=0}^{N-1} score_i[X]$, where X is a K-gram and score$_i$[X]=0 if X does not exist in $TFDB_i$.
2. Duplicate TTFDB and scoreT[] to site$_i$, i=1,2,…N-1.

3.2 Promoter Positioner(PP)

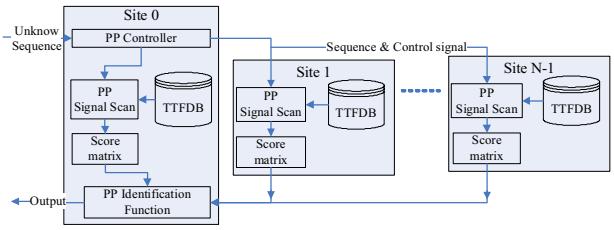

Fig. 4 PP's architecture

PP is also implemented by cluster system as shown in Fig.4. The Signal Scan partitions an unknown input sequence U of length L' into N segments as balanced as possible according to node performance, i.e., a longer segment is distributed to a higher performance node. U is virtually partitioned into units, each is 100 bps in length, and a segment, say i, of C_i+1 units starting from the $(\sum_{j=0}^{i-1} C_j)^{th}$ to $((\sum_{j=0}^{i} C_j)+1)^{th}$ unit is distributed to site$_i$ where $C_i=\left\lceil \left\lceil \frac{L'}{100} \right\rceil * \frac{Pc_i}{\sum_{i=0}^{N-1} Pc_i} \right\rceil$. Site$_i$ picks up segment i which overlaps 100 bps with segment i+1 as the test data. Site$_i$ partitions segment i into K-grams and generates a score matrix SM$_i$[][] with algorithm4:

Algorithm4: Generating a score matrix /* performed by sitei

Input: segment i, TTFDB.

Output: SMi[][].

Method:

1. Generate a score matrix SM$_i$[6 to 21][0 to M+99], where M= C_i*100 and the initial value of each element is 0.

2. For K=6 to 20
{Partition segment i, along its sequence, into K-grams.
For each K-gram, says KG, if KG ∈ TTFDB, give KG's score to corresponding SM entry i.e., $SM_i[K][q]$ = score[KG], where q is the q th bp of segment i.}
3. Send SMi[][] to $site_0$.

$Site_0$ collects $SM_i[][]$, i=0,1,2,…N-1 , appends $SM_{i+1}[][]$ to $SM_i[][]$ to form a complete score matrix SM as shown in Fig.5 and sums up score for each column t from SM[6][t] to SM[20][t] as a subtotal which is then stored in SM[21][t], $0 \le t \le L'-1$. Next, SM[21][] is partitioned into $P=\lfloor L'/100 \rfloor$ units, named l-unit (long unit), each is 200 bps in length starting from SM[21][x] to SM[21][x+199], where x=100*r, r=0,1,2…,P-1, i.e., l-unit i overlaps l-unit i+1 100 bps and the last q bps, only appears in the final unit beginning at 100*(P-1). Finally, 200 subtotals in each l-unit X are summed up as a score which is then stored in the Xth entry, US[X], of an array US[0 to P-1]. We discover that every pattern meets the situation US[s]>0, US[s+1]<0 and US[s+2]<0, US[s]'s corresponding subsequence (200 bps) possibly contains a promoter or a part of a promoter (a binding site). The processing details are described in Algorithm5.

	1	2	3	4	5	6	7	8		94	95	96	97	98	99	100	101	102	103		119	120	121	122	123	124	
sequence	c	a	t	c	a	t	c	g	---	a	t	t	c	c	t	c	g	t	c	---	g	c	a	c	t	t	---
Segment 6bp						1	1	1			1	1	1	-1	-1	1	-1	-1	-1			1	1	1	1	-1	-1	
Segment 7bp							1	1	---		1	1	1	1	-1	-1	-1	-1	-1	---		-1	-1	1	1	1	-1	---
Segment 8bp								0			1	1	1	1	-1	-1	-1	-1	-1			1	1	-1	1	1	1	
Segment 9bp											1	1	1	0	-1	-1	1	-1	-1			1	1	-1	1	0	1	
⋮									⋮		⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮			⋮	⋮	⋮	⋮	⋮	⋮	
Segment 19bp											0	1	0	0	1	-1	1	1	0	1		1	1	1	1	0	-1	
Segment 20bp											0	0	0	0	0	0	-1	-1	0	-1		0	0	1	0	0	0	
SCORE						1	2	2	---		9	14	11	4	-5	-8	-6	-7	-9	-3	---	6	4	6	8	3	-5	---

Fig. 5 A sample of SM[][] representing the segment is CATCATCG…ATTCC…

Algorithm5: Processing SMi /* performed by site0
Input: SMi[][], i=0,1,2…,N-1.
Output: Possible promoter regions (PPR).
Method:
1. Append SMi+1[][] to SMi[][] to form SM[][] and PPR=∅.
2. Calculate subtotal for each column of SM[][].
3. Sequentially segment SM[21][] into $\lfloor L'/100 \rfloor$ l-units, each is 200 bps in length and overlaps its next unit 100 bps.
4. For each l-unitX, calculate US[X], X=0,1,2,..,P-1.
5. For each (US[s]>0 and (US[s+1]<0 and US[s+2]<0)), $0 \le s \le P-3$
PPR=PPR $\bigcup$ {US[s]'s corresponding l-unit} as the possible promoter regions according to our previous assumption.

4 Experiments and Simulation

There are 2541 vertebrate promoters in the EPD Release 80. Some of them marked by "N" involve unknown nucleic acid. VPPS excludes those with "N". Therefore, only 2237 remain. We choose 2,000 and pick up 550 bps, including TSS's 500 upstream bps and 50 downstream bps both beginning their counting at TSS, from each. We also select 2,000 non-promoters from the GenBank, from each 550 bps are also extracted, from the same positions as those of promoters. The remaining 237 promoters and other 2000 non-promoters are extracted from the GenBank as the test data.

Two kinds of scoring methods are used. The first is counting the times a K-gram KG appears in test sequence. If KG appears in TTFDB and score[KG]>0, add score_TFBS[KG] by one. If score [KG]<0, reduce score_TFBS[KG] by one. The second method is based on score_TFBS[KG] = score_TFBS[KG] + score[KG] instead of increasing/decreasing by one. Each experiment is performed five times, each time different 2000 non-promoters for training and different 2000 for test are involved, and each time different 237 promoters are selected as the test set, the remaining 2000 are training set due to no more promoters. Each listed score is the average of its five independent test results. SET1 is obtained by using the second scoring method, while SET2 is generated by invoking the first. SET3 is the same as SET1, but only TFBSs/NTFBSs with scores within top 5% and bottom 5% rather than all TFBSs/NTFBSs are compared. SET4 is the same as SET2, however the TFBSs/NTFBSs compared are also those within top 5% and bottom 5%. Fig. 6 shows VPPS's true positives.

Since the experimental result of top (bottom) 5% is similar to that of top (bottom) 10% (only 0.3% difference) and the computation complexity of the former is one half of the latter, VPPS adopts top (bottom) 5%.

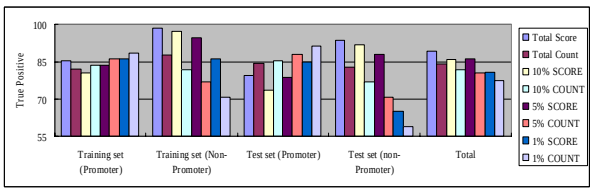

Fig. 6 True positives (TPs) of VPPS's experimental results

From Table 1, we can deduce two conclusions. First, the result obtained by accumulating scores has better performance than frequency. For example, in SET3,

the accuracies of training sets regarding promoter and non-

Table 1 The results of different training and test data sets

	Number of sequence	AVG true positive	Accuracy	Standard Deviation	AVG true positive	Accuracy	Standard Deviation
		SET1 (score,all):Training set			SET3 (score,5%):Training set		
Promoter	2000	1705	0.853	0.038	1671	0.835	0.084
Non-promoter	2000	1972	0.986	0.005	1894	0.946	0.044
		Test set			Test set		
Promoter	237	189	0.797	0.043	186	0.785	0.096
Non-promoter	2000	1874	0.937	0.024	1757	0.879	0.041
		SET2 (freq.,all):Training set			SET4 (freq.,5%):Training set		
Promoter	2000	1644	0.822	0.013	1724	0.862	0.039
Non-promoter	2000	1750	0.875	0.015	1538	0.769	0.081
		Test set			Test set		
Promoter	237	200	0.844	0.024	209	0.882	0.038
Non-promoter	2000	1652	0.826	0.039	1418	0.709	0.059

Table 2 performance comparison of relevant tools

			Accuracy	Training/Test set's accuracy	Total accuracy
NNPP	Training set	Promoter	0.741(1482/2000)	0.65 (2603/4000)	0.619 (3864/6237)
		Non-promoter	0.561(1121/2000)		
	Test set	Promoter	0.679(161/237)	0.563 (1260/2237)	
		Non-promoter	0.55(1099/2000)		
Promoter Scan	Training set	Promoter	0.502(1002/1997)	0.725 (2895/3993)	0.790 (4923/6229)
		Non-promoter	0.948(1893/1996)		
	Test set	Promoter	0.5(118/236)	0.906 (2028/2236)	
		Non-promoter	0.955(1910/2000)		
Promoter 2.0	Training set	Promoter	0.195(389/2000)	0.512 (2051/4000)	0.589 (3678/6237)
		Non-promoter	0.831(1662/2000)		
	Test set	Promoter	0.191(46/237)	0.747 (1672/2237)	
		Non-promoter	0.813(1626/2000)		
VPPS	Training set	Promoter	0.835(1671/2000)	0.892 (3565/4000)	0.883 (5508/6237)
		Non-promoter	0.946(1894/2000)		
	Test set	Promoter	0.785(186/237)	0.868 (1943/2237)	
		Non-promoter	0.879(1757/2000)		

promoter are 83.5% and 94.6% respectively. The corresponding results in SET4 are only 86.2% and 76.9% respectively. The accuracy of SET3 is about 9.9% (=((1671+1894+186+1757)-(1724+1538+209+1418)) /6237) better than that of SET4. Second, using the top and bottom 5 % TFBS/NTFBSs data is about 3.7%(=((5740-5508)/6237)) less than using all TTFDB data, but the processing efforts can be dramatically reduced since the search space is only 5% of TTFDB. The total true positives in SET1 are 1894 (1705+189) and 3846 (1972+1874), whereas those in SET3 are 1857 (1671+186) and 3651 (1894+1757). Also the lower standard deviation represents our algorithms and experiments are stable and reliable when different training and testing data are in use. Table 2 summarizes the test results of some relevant tools downloaded from the Internet, like NNPP, promoter scan, and promoter 2.0. We can conclude that most tools are better in predicting non-promoters. For example, with Promoter Scan the training set accuracies in predicting promoter and non-promoter are 50.2% and 94.8% respectively. In Promoter 2.0, the accuracies are 19.5% and 83.1% respectively. VPPS's are 83.5% and 94.6%. VPPS seems having better total performance. Furthermore, high true positive implies low false negative which means less genes are lost. This occurrence is particularly valuable in predicting promoters.

In cluster performance analysis, training data include 2000 promoters and 2000 non-promoters, each is 550 bps in length. A eight-site cluster computer is in use. Each site is equipped with intel 2.8G CPU and 1.5G RAM. The results are shown in Fig. 7 depicting that the performance is almost proportional to the number of sites but with some level of overhead.

In sequence prediction, input sequences of 1000, 5000, 10000, 20000, and 40000 bps in length and the eight-site cluster computer are involved. The results are shown in Fig. 8. We found that a longer sequence has a better performance due to less proportional overhead. The phenomenon is stronger as more sites are used.

Let $S(n) = \frac{\text{Number of computational cost using one processor}}{\text{Number of parallel computational cost with n processors}}$ be the speedup by using n processors over using only one processor. We can conclude from Fig. 9 that in VPPS a longer sequence with more cluster sites will result in better speedup.

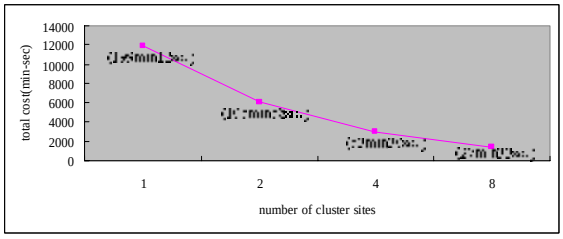

Fig. 7 training performance by using cluster computing

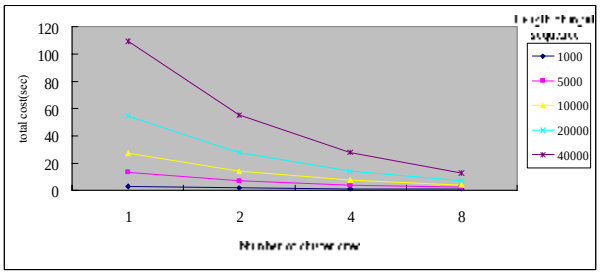

Fig. 8 recognition performance (sec)

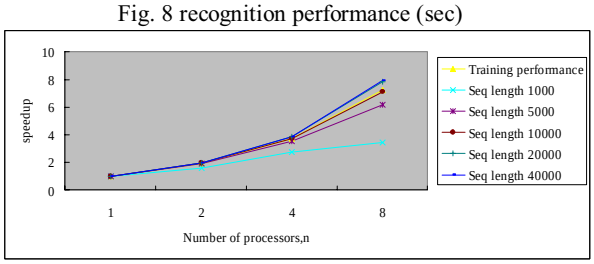

Fig. 9 Speedup against number of processors

5 Conclusion

In this article, we propose a system, VPPS, aiming to recognize promoter within an anonymous vertebrates DNA sequence. VPPS is general promoter prediction tool suitable for predicting sequences of different species, such as those of humans or mice. It can be applied to recognize the difference between intron sequences and promoter sequences that are found to be interesting.

The experimental results show that the prediction accuracies of VPPS have significantly improved when comparing to other reported web-accessible promoter recognition systems. VPPS is also able to more accurately point out the position of a possible promoter given a large anonymous DNA sequence, and to predict more promoters. This occurrence can significantly improve research result in investigating gene behaviors and make gene hunting easier since VPPS considerably reduces the possibility of false positive and false negative as comparing to other systems.

However, VPPS has its own bottleneck especially in training phase. It spends most cost, almost 95%, to parse K-grams and search database (TFDB and TTFDB). We solve this problem by deploying cluster computer to distribute and share workload. The experimental results show that VPPS successfully shortens prediction cost with the assist of cluster computing.

Furthermore, we have never involved any well-known promoter patterns, like TATA-box, Inr, CCAAT-box and CpG islands. Only simple statistic techniques are used. As those patterns are involved, its predictive accuracy should be able to successfully achieve to a higher step.

References

[1] Fickett, J.W. and Hatzigeorgiou A.G., "Eukaryotic Promoter Recognition," Genome Research, vol. 7, pp. 861-878, Sept. 1997.

[2] Leu. F.Y., Lin. J.C., Li. M.C. and Yang. C.T., "A Performance-Based Grid Intrusion Detection System," Proceedings of the 29th Annual International Computer Software and Applications Conference, pp.304-309, March 2005.

[3] Ohler, U., Harbeck, S., and Niemann, H., et al., "Interpolated Markov Chains for Eukaryotic Promoter Recognition," Bioinformatics, Vol. 15, pp. 362-369, May 1999.

[4] Pedersen, A.G., Baldi, P., Chauvin, Y., and Brunak, S., "The Biology of Eukaryotic Promoter Prediction-a Review," Computers and Chemistry, vol. 23, pp. 191-207, 1999.

[5] Périer, R.C., Junier, T., Bonnard, C., and Bucher, P., "The Eukaryotic Promoter Database EPD," Nucleic Acids Research, Vol. 26, pp. 353-357, Jan. 1998.

[6] Sudharshan Vazhkudai., "Enabling the Co-Allocation of Grid Data Transfers'," Proceeding of the Fourth International Workshop on Grid Computing (Grid'03) IEEE, pp.44-51, Nov. 2003.

[7] Werner, T., "Models for Prediction and Recognition of Eukaryotic Promoters," Mammalian Genome, vol. 10, pp. 168-175, 1999.

[8] NCBI-National Center for Biotechnology Information, http://www.ncbi.nlm.nih.gov/.

[9] Expression, genes & more glossary http://www.genomicglossaries.com/content/ex_bio.asp