International Journal of Biological, Life and Agricultural Sciences
ISSN: 2415-6612
Vol:5, No:10, 2011

A New Edit Distance Method for Finding Similarity in Dna Sequence

Patsaraporn Somboonsak, Mud-Armeen Munlin

***Abstract*—**The P-Bigram method is a string comparison methods base on an internal two characters-based similarity measure. The edit distance between two strings is the minimal number of elementary editing operations required to transform one string into the other. The elementary editing operations include deletion, insertion, substitution two characters. In this paper, we address the P-Bigram method to sole the similarity problem in DNA sequence. This method provided an efficient algorithm that locates all minimum operation in a string. We have been implemented algorithm and found that our program calculated that smaller distance than one string. We develop P-Bigram edit distance and show that edit distance or the similarity and implementation using dynamic programming. The performance of the proposed approach is evaluated using number edit and percentage similarity measures.

***Keywords*—**Edit distance, String Matching, String Similarity

I. Introduction

THE edit distance is a common similarity measure between two strings. It is defined as the minimum number of insertions, deletions or substitutions of single terminal needed to transform other of the strings into the other one. This distance is a key importance in several fields, such as Bioinformatics, Text processing consequently computational problems. Given a string $S_1 = [a_1\ a_2\ a_{3...}a_n]$ and $S_2 = [b_1\ b_2\ b_{3...}b_m]$ as the minimal cost of transforming S_1 into S_2 using the three operations insert, delete, and substitution, where only unit cost operations are considered in edit distance. The cost of elementary editing operations is given by some scoring function which induces a metrical on strings. The similarity of two strings is the minimum number of edit distance. DNA sequence can be seen as a pair of reverse complementary repeats in a string that are separated by a number of Nucleotide. The complementary relation on nucleotides (A T C G) means that A is complementary to T and C is complementary to G.In this paper, we develop two-letter edit distance. We compute the edit distance and the cost of operation with dynamic programming. The new algorithm is to find all minimal distance in a string.

II. Related Work

The early works on finding similarity in strings deal with edit distance. The edit distance models were studied in two contexts, for string matching and for sequence similarity. A lot of works have been on string matching (see [6-11]).

P. S. Author is with the Faculty of Information Science and Technology Mahanakorn University of Technology, Nongchock, 10530 Bangkok Thailand (e-mail: psomboonsak@yahoo.com).

M. M. Author, is with the Faculty of Information Science and Technology, Mahanakorn University of Technology, Nongchock, 10530 Thailand (e-mail: mmunlin@gmail.com).

We will focus on edit distance techniques as our main goal here is to concentrate on similarity. We decided to use the most common measure: the Levenshtein edit distance[4]. In the text is typically assumed to be random, i.e., each character is chosen uniformly and independently from the alphabet. A sequence alignment is a way of arranging the sequence of DNA, RNA or protein to identify regions of similarity that may be a consequence of functional, structural or evolutionary relationships between the sequences[2]. The first application of the edit distance algorithm for protein sequence was studied by Needleman-Wunsch[1].It is commonly used in bioinformatics to align protein or nucleotide sequences. To find the alignment with the highest score, a two-dimensional array. The algorithm is an example of dynamic programming matrix. There is one column for each character in sequence A, and one row for each character in sequence B. This algorithm progresses, the $F_{i,j}$ will be assigned to be the optimal score for alignment of the first i = 0,…n characters in A and the first j = 0,…m characters in B. This algorithm works in the same way regardless of length or complexity of sequences. The simplest and most common scoring function is the Levenshtein distance[4] which assigns a uniform score of string for every operation. Determining the edit distance between a pair of string is a fundamental problem in computer science in general, and in combinatorial pattern matching in particular, with applications ranging from database indexing and word processing, to bioinformatics[3][5]. The Levenshtein distance[4] between two strings is the minimum number of editing steps that convert one string into another. Given two string A[1…m] and B[1…n], one can calculate their edit distance by dynamic programming. We refer to matrix D[0..m, 0..n] as the edit distance table of string A and B. Initialy, D[i,0] = i for $0 \le i \le m$ and D[0,j] = j for $1 \le i \le n$. Then the cell D[i,j], where i,j > 0, stores the edit distance of string A[1…i] and B[1…j]. We also say that the cells D[i,j], where j - i = d are on diagonal d of D[15]. We introduce some important properties of the edit distance table that are constantly used in later discussion. The Longest common subsequence(LCS)[14] problem is to find the maximum possible length of a common subsequence of two strings, 'a' of length |a| and b of length |b|. The sequence similarity analysis is the Longest Common Subsequence(LCS) problem, where we eliminate the operation of insertions deletions and substitutions[14]. Given strings S and T of lengths n and m, respectively, over an alphabet Σ, determine the lengths of the longest subsequence that is common to both s and t. Here, s subsequence of $S = s_1s_2,…s_n$ is a string of the form $si_1\ si_{2,...}\ si_k$, where each i_j is between 1 and n and $1 \le i_1 < i_2 ,…< i_k \le n$. For example, if Σ = {A, T, C , G}, S = AGCGA and T = CAGATAGAG, then CGA is a subsequence of length 3 of both S and T. However, it is not the longest common

subsequence of S and T, since the string AGGA is also a common subsequence of length 4 of both S and T. Since these two strings do not have a common subsequence of length of the longest common subsequence of S and T is 4[16].Bioinformatics have developed several means to characterize the similarity between genetic sequences. One intuitively appealing measure is the edit distance. The edit distance was originally proposed by Levnshtein[4]. One of the major problems is that of edit distance. To compare two strings, a common technique is so called string edit distance, as a measure of their dissimilarity. Edit distance is equal to the minimum number of editing operations required to transform one sequence into the other. The three basic editing operations are insertion, deletion, and substitution[11]. The operations to transform one string into anther string using the basic character wise operation delete, insert and substitution. If each operation has cost of 1, then edit distance is the number of operations. The minimal number of these operations is called edit distance or Levenshtein[4] distance. Each operation has an associated cost, which is a function of characters involved in operation. The cost of transformation is the sum of cost the individual operations. Dynamic programming (DP) is a method used for optimizing a multistage process and that is particularity applicable to problems requiring a sequence of interrelated decisions. Each decision transforms the current situation into a new situation. A sequence of decisions, which in turn yields a sequence of situations, is sought that minimizes some measure of value. The value of a sequence of decisions is generally equal to the sum of the values of the individual decisions and situations in the sequence. It is a "solution seeking" concept which replaces a problem of n decision variable. Such an approach allows analysts to make decisions stage by stage, until the final result is obtained. We use the following formulation[13].Let $S = [s_1,s_2 \ldots s_m]$ be the source sequence, $T = [t_1,t_2 \ldots t_n]$ the target sequence, and $d_{i,j}$ the distance between the subsequence $[s_1,s_2,\ldots s_i]$ and $[t_1,t_2 \ldots t_j]$;

Then for $1 \le i \le m$, $1 \le j \le n$,

$$d_{0,0} = 0$$
$$d_{i,0} = d_{i-1,0} + c(S_i,\emptyset),$$
$$d_{i-1,j-1} = d_{j-1,0} + c(\emptyset,t_j),$$

and

$$d_{i,j} = \min \begin{cases} d_{i-1,j} + c(S_i,\emptyset), \\ d_{i-1,0} + c(\emptyset,t_j), \\ d_{j-1,0} + c(S_i,t_j), \end{cases}$$

where $c(S_i,\emptyset)$ is the cost of deleting S_i, $c(\emptyset,t_j)$ is the cost of inserting t_j, and $c(S_{i,tj})$ is the cost of substituting t_j for s_i. The edit distance between S and T is simply $d_{m,n}$.

III. Materials and Methods

In this section we describe the algorithm for finding P-Bigram similarity new edit distance. We first describe P-Bigram Distance, following which we describe reduced operation to achieve a time and space efficient algorithm. The reduce use similar idea to those in Levenshtein[4] distance.

This P-Bigram edit distance is an importance of the edit distance for two characters which is also used for measuring the similarity.

A. P-Bigram Edit Distance

We define a P-Bigram Edit Distance as a character that is present in two nucleotide sequences. So, we estimate the minimum length for which a maximal match is significant, according to the length of the two compares sequences. Let $S = \{s_1,s_2,s_3,\ldots s_n\}$ be a set of n characters and $T = \{t_1,t_2,t_3,\ldots t_m\}$ in a text of m characters which are strings of nucleotide sequence characters of length i and j over the nucleotide alphabet S = {A, C, G, T}.

Definition 1. A P-Bigram Edit distance solution for computing the edit distance between a pair of string $S = s_1,s_2 \ldots s_N$ and $T = t_1,t_2 \ldots t_M$ involves filling in an (n+1) X (m+1) table P, with P[i,j] sorting the edit distance between $s_1,s_2 \ldots s_i$ and $t_1,t_2 \ldots t_j$. In addition let |S| and |T| denote the length of string S and T. We consider the Levenshten edit distance. The computation is done according to the base-case rules given by P[0,0] = 0, P[i,0] = T[i-1,0] + cost of deleting s_i, and P[0,j] = P[0,j-1] +1 cost of inserting t_j, and according to the following dynamic programming step:

$$P[i,j] = \min \begin{cases} P[i-1,j] + \text{the cost (1) of delete } s_i \\ P[i,j-1] + \text{the cost (1) of insert } t_j. \\ P[i-1,j-1] + \text{the cost (1) of substitute } s_i \text{ with } t_j \end{cases}$$

Definition 2. For a string S and value of $\in$, let $P = \in |S|$ and string T and value of $\in$, let $P = \in |T|$. For example, the two-letter alphabet {s, t}, if S = ATTCCGGTCAAG and T = ATTGGTTCCAAGGA. The first scans and match two-letter of sequence from left to right then a pair of string S is{ AT, TC, CG, GT, CA, AG } and a pair of- string T is { AT, TG, GT, TC, CA, AG, GA }. The P-Bigram Edit Distance between two characters with two strings. Given two strings S, T a standard technique for computing P-Bigram (S, T) is the dynamic programming method, where we compute the DP matrix of size (|S| + 1) for which DP [i,j] = P-Bigram (S [1..i]), T[1..j]) for $1 \le i \le |S|$ and $1 \le j \le |T|$.

The compute matrix is the following:

$$DP[i,j] = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0, \\ \max DP[i-1, j], DP[i, j-1]) & \text{if } i,j > 0 \text{ and } S[i] \neq T[j] \\ DP[i-1, j-1] + 1 & \text{if } i,j > 0 \text{ and } S[i] = T[j] \end{cases}$$

Therefore, to compute P-Bigram = DP S and T of length |S| and |T| with a time complexity of O(| S| X |T|) and space complexity of O(min(| S| X |T)).

ALGORITHM I
P-BIGRAM EDIT DISTANCE

```
1  EditDistance S(0…s-1) T(0…t-1)
2     int   m[i,j] = 0
3     for   i ← 0 to n
4     do    m[i,o] = i
5           i = i + 1
6     for   j ← 0 to m
7      do   m[o,j] = j
8           j = j + 1
9     for   i ← 0 to n;  i ++
10    for   j ← 0 to m;  j ++
11       if s[i,i-1] = t[j,j-1] then
12          cost ← 0
13           m[i,j] ← m[i,j] +1
14           opt  ← delete(i,n)  // opt operation  of  i and j
15       else
16          if s[i,i-1] ≠ t[j,j-1] then
17          cost ← 1
18           m[i,j] ← m[i,j] +1
19           opt  ← insert(n,i,j)  // opt operation  of  i and j
20       else
21          if s[i,i-1] ≠ t[j,j-1] then
22          cost ← 1
23           m[i,j] ← m[i,j] +1
24           opt  ← substitution(n,i,j)  // opt operation  of  i and j
25     for i ← 1 to |S|
26      do for  j ← 1 to |T|
27         do m[i,j] = min{m[i-1,j-1]
28                        m[i-1,j] + 1,
29                        m[i,j-1] + 1}
30    return m[ n,m]
```

P-Bigram algorithm compute the edit distance. Suppose we wish to calculate the edit distance between the strings S = $s_1,s_2…s_n$ and T = $t_1,t_2…t_m$.

1. We begin by forming an (n + 1) X (m + 1) matrix P initially containing all zeros, that is $P_{(i,j)} = 0$ for i = 0,1,2,…,n and for j = 0,1,2,…,m.
2. Assign values P_{0j} = j, j = 0,1,2,…,m and P_{0i} = i, i = 0,1,2,…,n.
3. Starting from the second top row and going from left to right, we fill in the values $P_{i,j}$ according to the following step:

$$\left\{\begin{array}{l} \text{Insertion} = P_{i-1,\,j-1} + \text{cost} \quad \text{Deletion} = P_{i-1,\,j} + 1 \\ \text{Substitution} = P_{i,\,j-1} + 1 \quad P_{i,j} = \min(\text{substitution}, \\ \text{deletion , insertion}). \end{array}\right\}$$

   Where cost = 0 if $s_i = t_j$ and cost = 0 if $s_i \neq t_j$.
4. After computing a row, move to the row below, until the bottom row is reached.
5. The value P_{nm} is the edit distance between the string S and T.

Example Here is an example of computing the P-Bigram edit distance of two characters in two strings.

	T_j	AT	TG	GT	TC	CA	AG	GA
S_i	0	1	2	3	4	5	6	7
AT	1	0	1	2	3	4	5	6
TC	2	1	1	2	2	3	4	5
CG	3	2	2	2	3	4	5	6
GT	4	3	3	2	3	4	5	6
TC	5	4	4	3	2	3	4	5
CA	6	5	5	4	3	2	3	4
AG	7	6	6	5	4	3	2	3

Fig. 1 P-Bigram edit distance of S_i = ATTCCGGTCAAG and T_j = ATTGGTTCCAAGGA.

The values of the above table have been obtained with the following unitary costs(Fig. 1, Fig. 2) Sub(s, t) = 1 if s ≠ t and Sub(s,t) = 0, Insert(t) = Delete(t) for s,t ∈ P.

A score (instead of a cost) is associated with each elementary edit operation. For s,t ∈ P:

- Sub(s,t) denotes the score of substituting the character t for the character s,
- Del(t) denotes the score of deleting the character t,
- Ins(t) denotes the score of inserting the character t.

	0	1	2	3	4	5	6
	1	1	2	2	3	4	5
S	2	2	2	3	4	5	6
	3	3	2	3	4	5	6
	4	4	3	2	3	4	5
	5	5	4	3	2	3	4
	6	6	5	4	3	2	3
			T				

Fig. 2 The DP matrix for P-Bigram(S,T), where S = ATTCCGGTCAAG and T = ATTGGTTCCAAGGA

The main idea behind the P-Bigram edit distance described above is that each entry $P_{i,j}$ corresponds to the minimal number of editing operations required to transform the substring $S_i = s_1,s_2…s_i$ into the substring $T_j = t_1,t_2… t_j$. Initially, an empty string to transformed into a string of k characters by using exactly k additions (Step2). Explanation of Step3:

- If we can transform S_i into T_{j-1} in $P_{i,j-1}$ operations, then we can transform S_i into T_j in insertion = $P_{i-1,j-1}$ + 1 operations by simply adding the characters s_i to T_{j-1}.
- If we can transform S_{i-1} into T_j in $P_{i-1,\,j}$ operations, then we can transform S_i into T_j in deletion = $P_{i-1,j}$ + 1 operations by simply deleting the characters s_i from to S_i.
- If we can transform S_{i-1} into T_{j-1} in $P_{i-1,j-1}$ operations, then we can transform S_i into T_j in substitution = $P_{i,j-1}$ + 1 cost operations by simply substitution the characters s_i with T_j if they are different (cost = 1).

- The minimal number of operations required to transform T_i into S_j is the minimum of the three quantities: P_{ij} = min(substitution, deletion , insertion).

IV. Evaluation And Results

In this section, we want to evaluate a similarity measure, so our evaluation will focus on edit distance a set of source nucleotide to a set of target nucleotide. We suppose that the size of the target set is a part of nucleotide to contain possible other similar nucleotides. The P-Bigram Edit Distance task consist of comparing two characters with two strings that contain nucleotide virus in order to decide whether two strings refer to the same entity. A data set used for testing the nucleotide virus dynamic programming techniques is usually represented as two sets of strings and subset of their Cartesian product that defines valid matches. We measured percentage similarity and edit of the matched characters in the string. It follows from definitions that the following (Table 1).

Table I
Notation

sim	Similarity between S and T is related to their commonality
$sim_{edit}(x,y)$	Similarity of edit
editDist(x,y)	Minimum number of character (insertion, deletion, substitution)

Several metrics to evaluate effectiveness of character identification techniques have been proposed, combining such criteria. The problem of DNA similar measure is a different measures how many match is identified in relation to the total number of edit. The Results given in Table1 are the percent similarity and number of edit[12][17].

$$sim_{edit}(x,y) = \frac{1}{1 + editDist(x,y)}$$

Where editDist(x,y) is the minimum number of data(DNA) insertion deletion and substitution operations needs to transform one string to the other.

Example Here is an example of computing the P-Bigram similarity of two characters in two strings(ATTCCGGTCAAG, ATTGGTTCCAAGGA).

$$sim_{edit}(s,t) = \frac{1}{1 + 3(s,t)}$$

$$= 0.25$$

Similarity = 25.00%

Our training set is an export of the NCBI(National Center for Biotechnology Information) data. To simplify the evaluation, we had to set a threshold to decide if a nucleotide is a small edit operation. Using our trained similarity measure we computed the similarity between two characters in two strings and look for minimum edit distance (similarity) that was able to reduce edit operations. Our evaluation concentrates on trying to match two characters nucleotide. We summarize the results using edit and similarity in Table1, Table II, Table III, Figure 4, Figure 5.

Table II
Number edit for pair of DNA sequence

Source	Destination	LD	LCS	ND	PB
		Edit	Edit	Edit	Edit
ATTCCG GTCAAG	ATTGGTT CCAAGGA	6	10	7	3
GAATTC AGTTA	ATTGGTTC CCAAGGA	5	6	8	4
GCATCG GTAATT	ATCTCG GACG	7	6	6	5
GCCCTA GCG	GCGCAA TG	4	5	8	3
TGATCG ATC	CTGATCG ATC	1	9	7	1

By using the edit distance we find the possible edit of sequence. Now to edit distance in many algorithms we used to compare edit. A better approach is to edit in such a way which minimum edit.

Table III
Similarity for pair of DNA characters in many algorithm

Source	Destination	LD	LCS	ND	PB
		Similarity	Similarity	Similarity	Similarity
ATTCCG GTCAAG	ATTGGTT CCAAGGA	14.29%	9.09%	12.50%	25.00%
GAATTC AGTTA	ATTGGTTC CCAAGGA	16.67%	14.29%	11.11%	20.00%
GCATCG GTAATT	ATCTCG GACG	12.50%	14.29%	14.28%	16.67%
GCCCTA GCG	GCGCAA TG	20.00%	16.67%	11.11%	25.00%
TGATCG ATC	CTGATCG ATC	50.00%	10.00%	12.50%	50.00%

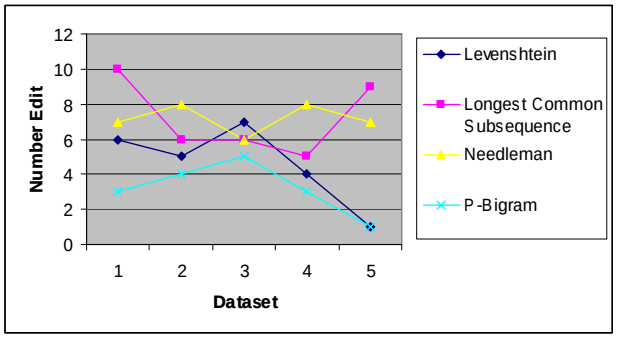

Fig. 4 Number Edit

Fig. 4 Show the comparisons of different algorithms with the sim_{edit} metrics. The current metrics gives good performance in reducing the number of editions compared with other popular methods.

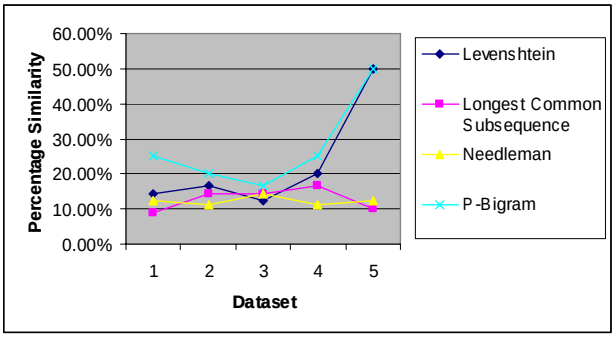

Fig. 5 Comparison of different algorithms with percentage similarity

Fig. 5 The cross line shows the P-Bigram algorithm where as longest common subsequence algorithm and Needleman algorithm and Levenshtein shown by square, triangle, diamond. The P-Bigram Edit distance is a cheap distance measure which always returns a distance rather smaller than the Unigram edit distance, there are shown for typical results in Fig. 4 and Fig. 5. For given string S and T this distance can be calculated in $O(n^2)$. The P-Bigram Edit distance is then divided to this transformation the P-Bigram distance similarity measure will always be smaller than the edit distance similarity measure.

V. Conclusion

We proposed an edit distance of the P-Bigram method integrating the dynamic programming concept to compared similarity. That edit distance implements a heuristic of giving grater importance in the combined measure to the pairs of strings whose similarity is higher in comparison with the similarity of other pairs. The proposed method was tested on 5 dna-matching datasets with representative two characters-based string measures: Levenshtein distance. The result showed that the performance of the P-Bigram method can be improved. The result of such process can then compute similarities between two characters two nucleotide with quit high accuracy. However there are several improvements we will try to address in the future.

Acknowledgment

This works was financially supported by Faculty of Information Science and Technology Mahanakorn University of Technology.

References

[1] Saul B. Needleman, Christlan D. Wunsch, "A General Applicable to the Search for Similarities in the Amino Acid Sequence of two Proteins", J Mol. Biol, Vol. 48, 1970, pp. 443-453.
[2] David Sankoff, "Simultaneous solution of the RNA folding alignment and protosequence problems", Siam J. Appl Math, Vol. 45, 1985, pp. 810-825.
[3] A drien Coyette, et al., "Trainable Sketch Recognizer of Graphical User Interface Design", International Federation for Information Processing, Vol. 1, 2007, pp. 124-135.
[4] Levenshtein V.I. "Binary codes capable of correcting deletions, insertions and reversals", Soviet Physics Doklady, Vol. 8, 1966, pp. 705-710.
[5] Gusfield. "Algorithms on String Trees and Sequences", Computer science and Computational Biology Cambridge University Press, 1997.
[6] Hall, P.A.V, Dowling, G.R. "Approximate string matching", ACM Computing Surveys, Vol. 4, 1980, pp. 381-402.
[7] Christen, P. "A Comparison of Personal Name Matching Techniques and Practical Issues", Technical Report TR-CS-06-02, Joint Computer Science Technical Report Series, Department of Computer Science, 2006.
[8] Cohen, W. Ravikumar, P. Fienberg, S."A comparison of string distance metrics for name-matching tasks", IJCAI Workshop on Information Integration on the Web, Acapulco, Mexico, 2003, pp. 73-78.
[9] AbdulJaleel, N. Larkey, L.S. "Statistical transliteration for English-Arabic cross language information retrieval", CIKM, 2003, PP. 139-146.
[10] Linden, K, "Multilingual Modeling of Cross-Lingual Spelling Variants spelling variants Information Retrieval", Vol. 3, 2006, pp. 295-310.
[11] Ristad, E.S. Yianilos, P.N. "Learning string-edit distance", IEEE Transactions or Pattern Analysis and Machine Intelligence, 1998.
[12] Carlo Batini, Monica Scannapieco, "Data Quality Concepts, Methodologies and Techniqes", Springer, DCSA, 1998, pp. 117-127.
[13] Heikki Hyyrö, Ayumi Shinohara, "New Bit-Parallel-Distance Algorithm", Nikoletseas, LNCS 3503, 2005, pp. 380-390.
[14] Adrian. Horia Dediu, et al., "A fast Longest Common Subsequence Algorithm for Similar Strings", Language and Automata Theory and Applications, International Conference, LATA, 2010, pp. 82-93.
[15] Heikki Hyyrö, "Restricted Transposition Invariant Approximate String Matching Under Edit Distance", SPIRE, LNCS 3772, 2005, pp. 256-266.
[16] M.H. Alsuwaiyel, "Algorithms design techniques and analysis", World Scientific Connecting Great Minds, Vol. 7, 1999, pp. 203-208.
[17] Dekang Lin(1998), "An Information Theoretic Definition of similarity", Proceedings of the 15th international conference on statistic", Citeseer.

Mud-Armeen Munlin holds a Ph.D. degree in Computer Science from The University of Leeds, UK in 1995. Assistant Professor worked as associate Dean for Academics Faculty of Information Science and Technology Mahanakorn University of Technology Thailand. His specialized in solid modeling, virtual reality, medical application, internet and mobile applications.

Patsaraporn Somboonsak recrived the Master degree in Management Information Technology from Walailak University Thailand in 2006. Currently she is a Ph.D. student at Faculty of Information Science and Technology Mahanakorn University of Technology Thailand. Her interesting research are bioinformatics and computational for health medicine.