Journal of Computer Science 4 (6): 467-473, 2008
ISSN 1549-3636

Development of Deduced Protein Database Using Variable Bit Binary Encoding

B. Parvathavarthini, B. Rajesh Kanna and L. Rajeswaridevi
Department of Computer Application, St. Joseph's College of Engineering,
Chennai-119, TamilNadu, India

Abstract: A large amount of biological data is semi-structured and stored in any one the following file formats such as flat, XML and relational files. These databases must be integrated with the structured data available in relational or object-oriented databases. The sequence matching process is difficult in such file format, because string comparison takes more computation cost and time. To reduce the memory storage size of amino acid sequence in protein database, a novel probability-based variable bit length encoding technique has been introduced. The number of mapping of triplet CODON for every amino acid evaluates the probability value. Then, a binary tree has been constructed to assign unique bits of binary codes to each amino acid. This derived unique bit pattern of amino acid replaces the existing fixed byte representation. The proof of reduced protein database space has been discussed and it is found to be reduced between 42.86 to 87.17%. To validate our method, we have collected few amino acid sequences of major organisms like Sheep, Lambda phage and etc from NCBI and represented them using proposed method. The comparison shows that of minimum and maximum reduction in storage space are 43.30% and 72.86% respectively. In future the biological data can further be reduced by applying lossless compression on this deduced data.

Key words: Binary tree, protein sequence, amino acid

INTRODUCTION

RNA sequences are composed of four nucleotides: adenine (A), uracil (U), guanine (G), and cytosine (C). Any of the three combinations of the nucleotide bases is called as triplets or CODONS. Hence there are 64 possible CODONs. The combination of these CODONs forms different proteins. The sequence of amino acids represents a particular gene or protein [7]. The protein databases are generally represented in shorthand, using single letter designations. In the existing Protein database, the amino sequences are in a text format and are stored in any one of the following file formats such as flat, XML and relational files. This representation requires more memory storage spaces[3] and manipulation on these sequences can be dealt with high level programming languages. Hence it requires a high computation cost and high execution speed.

MATERIALS AND METHODS

Some of the existing universal biological databases details are listed below.

Universal protein sequence database: There are different categories of biological databases such as nucleic acid sequence, protein sequence and protein structure. Swiss-Prot[2] provides a biological database with a minimum level of redundancy and a high level of integration with other databases. The Protein Research Foundation Sequence Database[1] is the database of protein primary structures. The AceDB is an integrative database system that has been used for management of genome-oriented biological data. The Protein Data Bank (PDB) is the single world wide archieve of structural data of biological macromolecules. The generated sequence data are stored in large genomic repositories, of which the most commonly used databases are EMBL/European Bioinformatics Institute (EBI) and (NCBI)[2]. The Basic Local Alignment Search Tool (BLAST) is used to compare a novel sequence with those contained in nucleotide and protein databases[5,6]. ClustalW is a general purpose multiple sequence alignment program for DNA or proteins. The data storage formats used in different existing databases is shown in Table 1.

Types of data model: The bio informatics databases are maintained by different organizations using different DBMS with different data models such as flat files or XML or relational or object oriented[4]. Figure 1 shows the biological data storage in Flat, XML and RDB file format.

Flat file: A flat file database is the simplest database model in which the records are stored in one record per line format. Flat-file libraries contain data structured in an ASCII text is shown in Fig. 1a. The ASCII is the de

Corresponding Author: B. Parvathavarthini, Department of Computer Application, St. Joseph's College of Engineering, Chennai-119, TamilNadu, India

facto standard for data exchange (e.g., BLAST, CLUSTALW etc.)[4].

XML file: XML is a hierarchical and semi-structured model that has text-based files. An XML databank that stores data as a structured text file using XML tags (e.g., PIR-PSD) is shown in Fig. 1b.

Table 1: Various databases, their description, storage and data type

Database Name	Description of data	Data storage type	Data type
ACeDB	Seequence and variants	Object-oriented	Text, Numeric
PDB	Structure	Oracle	Blob, varchar2
BLAST	Sequence, analysis	FASTA	Text, Numeric
ClustalW	Sequence, analysis	FASTA	Text, Numeric

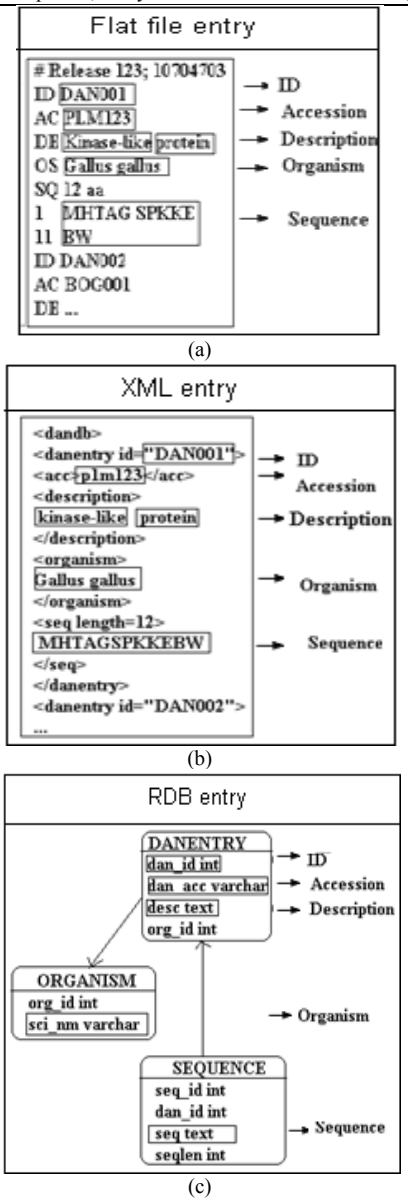

Fig. 1: Structure of a flat-file, XML and RDB entry

Relational database file: A Relational database file is a highly structured model. SQL statements are used to retrieve information from the database (e.g., AceDB). The results of the query are converted into standard text format. Figure 1c shows the storage format of relational database entry.

Iskandar *et al*.[9] stated software based approaches like navigational, mediator and data warehousing to integrate the different databases. All the file types discussed above have been used fixed byte representation to store amino acid sequences. But, our method uses variable bit to encode the data which reduces the storage space and it leads to explore the hardware for searching.

Proposed work: All existing protein databases use the ASCII code to encode each character. To encode a single amino acid, it requires 7 bits. Let n be the number of amino acids present in the protein. Then the actual memory needed to store the protein is 7n. To reduce the size of this protein database, a new approach is proposed here. This approach replaces the existing protein databases from the ASCII to a machine readable binary format using a probability-based variable bit length encoding technique. The proposed method reduces the storage space ranges from 43.85 to 87.17%. The detailed description about this reduction factor is described in result and discussion. Here, unique bits of binary codes are assigned to each amino acid sequence. This unique bit of binary codes is generated using a binary tree. The construction of this binary tree is discussed in section 3.2.2. For every amino acid, a probability value is calculated based on the occurrence of CODONs mapping a single amino acid. In this method, a lesser number of bits are assigned to amino acids which appear frequently and more number of bits to those which appear less frequently. The binary code representation of amino acids can be obtained using three techniques, namely, fixed length encoding, unique bit pattern-variable length encoding and probability based variable bit length encoding.

Fixed length encoding: By using fixed length encoding, each of 21 amino acids is represented as a bit pattern of fixed size 5 ($2^5 = 32$). Some amino acids occur more frequently than others. But all are assigned with the same number of bits. This results in a lengthy encoded sequence. Hence a variable length encoding scheme is proposed below to overcome this issue.

Variable bit length encoding: Variable bit length encoding is done, either by assigning a unique bit pattern to an amino acid or assigning the bit pattern based on the probability of occurrence of CODONs mapping a single amino acid.

Unique bit pattern variable BIT length encoding: If each amino acid is represented by a unique bit pattern(R = 0 L = 1 S = 00 P = 01 V = 10 A = 11 G = 000 T = 001 I = 010 O = 011 H = 100 K = 101 F = 110 Y = 111 N = 0000 D = 0001 C = 0010 Q = 0011 E = 0100 M = 0101 W = 0110), it is distinguishable only when presented separately. The difficulty arises, when these amino acids are formed into a data stream (00101010011110). Without a predictable bit-length, there is a chance of misinterpreting the code. This proposed algorithm, known as probability based variable bit length encoding, solves this issue.

Probability based variable length encoding: In this method, a probability value is calculated based on the frequency of the occurrence of the amino acid. The probability value is calculated using Eq. 1. These estimated probability values are assigned to the leaf nodes of a binary tree to be constructed. After the binary tree construction, a value zero is assigned to the right edges, and one, to the left edges of the tree. Then the binary code representation of an amino acid is identified by traversing the tree from the root followed by the branches that lead to that amino acid. Figure 2 shows the overview of the probability based variable length encoding process.

Calculation of probability value: Among the 64 possible CODONs, the number (N) of triplet CODONs needed to map a single amino acid is identified. Amino acids which have the same number of triplet CODONs are grouped. The Probability (P) of occurrence of each amino acid group appearing in the total combinations is identified using the Eq. 1.

$$P = \frac{N * C}{64} \quad (1)$$

where 'N' is the number of triplet CODONs needed to map a single amino acid and 'C' is the count of amino acid that has the same number of triplet CODONs. The probability value thus calculated for each amino acid group is shown in Table 2. The relative probability values assigned to each amino acid are given in Table 3.

Process for building the binary three based on the probability value: The process for building the binary tree is explained in the following steps.

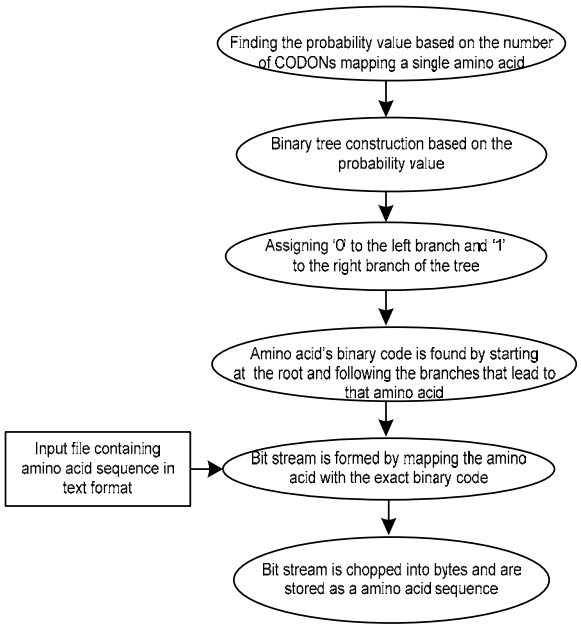

Fig. 2: Overview of probability based variable length encoding

Table 2: Probability value calculation

Amino acids (A)	No. of CODONs (N)	Count of A (C)	Probability value (P)
W,M	1	2	0.03125
E,Q,C,D,N,Y,F,K,H	2	9	0.28125
O,I	3	2	0.09375
T,G,A,V,P	4	5	0.3125
S.L.R	6	3	0.28125

Step 1: Organize the entire amino acid character set into a row, ordered according to its probability value from highest to lowest (or vice versa). Each amino acid character is now a node at the leaf level of a tree.

Step 2: Find the two nodes with the smallest combined probability value. Join them to form a single node that results in a two-level tree so that the combined two original nodes are the children of the new node. This node, one level up from the leaves, is eligible to be combined with other nodes. The sum of the weights of the other two nodes chosen must be smaller than the combination of any other choices.

Step 3: Step 2 is repeated until all the nodes, on every level, are combined into a single binary tree.

The Fig. 3 shows the leaf-level nodes representing the original probability values of amino acids arranged in the ascending order of value. The two nodes with the

Table 3: Symbol table to map amino acids in binary code

Amino Acid	CODON	Probability value	Binary code	Bit position has value 1	Binary code length
Tryptophan/W	UGG	0.015625	000000	-	6
Methionine /M	AUG	0.015625	000001	6	6
Glutamine/E	CAA,CAG	0.03125	00001	5	5
Glutamic acid/Q	GAA, GAG	003225	00100	3	5
Cysteine/C	UGC, UGU	0.03225	00101	3,5	5
Aspartic acid/D	GAC, GAU	0.03250	00110	3,4	5
Asparagine/N	AAC, AAU	0.03250	00111	3,4,5	5
Tyrosine/Y	UAC, UAU	0.03	10000	1	5
Phenylalanine/F	UUC, UUU	0.03	10001	1,5	5
Lycine/K	AAA, AAG	0.03025	10010	1,4	5
Histidine/H	CAC, CAU	0.03025	10011	1,4,5	5
Stop/O	UAA,UAG UGA	0.046875	1100	1,2	4
Isoleucine/I	AUA,AUC,AUU	0.046875	1101	1,2,4	4
Threonine/T	ACA,ACC,ACG,ACU	0.0625	0001	4	4
Glycine/G	GGA,GGC,GGG,GGU	0.0630	0100	2	4
Alanine/A	GCC,GCA,GCG,GCU	0.0635	0101	2,4	4
Valine/V	GUA,GUC,GUG,GUU	0.0615	1010	1,3	4
Proline/P	CCA, CCC, CCG, CCU	0.0620	1011	1,3,4	4
Serine/S	UCA,UCC,UCG,UCU,AGC,AGU	0.09365	0110	2,3	4
Leucine/L	UUA,UUG,CUA,CUC,CUG,CUU	0.09370	0111	2.3.4	4
Arginine/R	AGA,AGG,CGU,CGA,CGC,CGG	0.09390	111	1,2,3	3

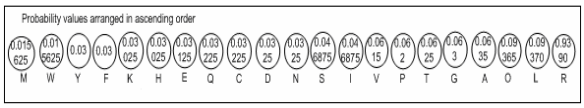

Fig. 3: Probability value arrangement

least probability values are identified and combined, which results in the creation of a new node.

The second row shows the next level of combined nodes. The probability value of the new node is found to be the sum of the two least values among the list of values. This decision keeps the branch lines crossing the tree. Hence the nodes are rearranged for clarity.

Assigning the code: The bit values are assigned to each branch of the constructed binary tree in such a way that the left of each node is assigned with 0 bit and the right of each node is assigned with 1 bit. The bit code assignment for the whole tree is shown in Fig. 4. The bit representation of any amino acid is found by the bits of the root followed by the branches leading to the leaf of that amino acid. The binary code for each amino acid is shown in Table 3. Using this probability-based variable length encoding technique, the binary code for each amino acid along with code length and the position containing bit value 1 is chosen in such a way that no code is the prefix of another code. Hence, there is no chance of misinterpreting the code.

Implementation of variable bit length encoding: This variable length encoding method maps an amino acid to a binary code and vice versa without any ambiguity.

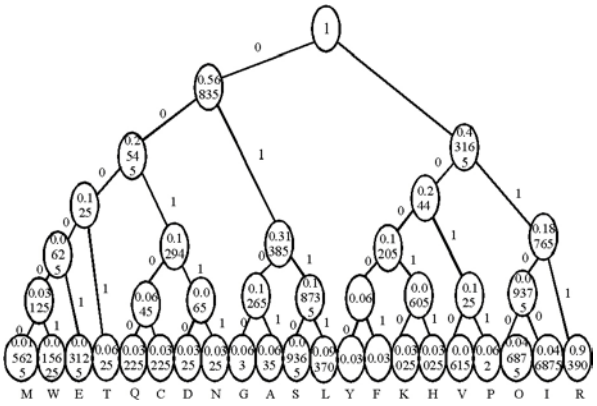

Fig. 4: Code assignment of amino acids

Now, this binary code representation of an amino acid is to be stored in memory, so that the resultant protein database is a deduced one. The technique used to store this binary code representation of an amino acid sequence is explained below.

Encoding algorithm: This encoding algorithm stores the binary code of an amino acid in byte form. Initially the bytes required to store an amino acid sequence are calculated from the code length stored in the symbol table, Table 3 and the memory space is allocated. From the symbol table, Table 3, the length of the binary code and the position containing the bit value 1 can be found. The pseudo code of the encoding algorithm is given below. The function and input variables in this encoding algorithm are also given below.

function Encoding (proteinDB, symbolTable [aminoAcid, size, position])
returns deduced proteinDB
Inputs: proteinDB, Protein database that contains amino acid sequence symbolTable, Contains amino acid character, size of the binary code, position containing 1 in the binary code
Local variables: Ptr = 0, sizeinbit = 0, sizeinbyte = 0, binaryByte [], c = 0
foreach aminoAcid in proteinDB do
sizeinbit = sizeinbit + aminoAcid.symbolTable.size
endfor
c = sizeinbit modulo 8
sizeinbyte = (sizeinbit + c)/8
The memory space of size sizeinbyte is allocated. i.e, binaryByte[sizeinbyte]
foreach aminoAcid in proteinDB do
foreach position of aminoAcid in symbolTable
do
binaryByte[aminoAcid.position + ptr] = 1
endfor
ptr = ptr + aminoAcid.size
endfor
return binaryByte
endfunction

RESULT AND DISCUSSION

The space needed to store an amino acid sequence using the proposed variable length encoding is compared here with the existing databases that are in the ASCII format. The space occupied by the existing protein database is equal to the product of the number of amino acids (n) in that sequence and seven. This comparison is carried out in three cases, namely, the worst case, the best case and the rest.

In the best case, the selected amino acid sequence contains R (Arginine) alone. Then, the size of the deduced protein database file created by the proposed method is 3(n-1)+6 including the start CODON. Hence the reducing factor is 87.17%. In the worst case, the selected amino acid sequence contains only W (Tryptophan) and M (Methionine). Then, the size of the deduced protein database file created by the proposed method is 6n [6(n-1)+6]. Hence the reducing factor is 42.86%.

$$\text{size} = \frac{\text{count(R)}}{n} + 4\sum_{i=1}^{8}\frac{\text{count(i)}}{n} + 5\sum_{j=1}^{9}\frac{\text{count(j)}}{n} + 6\sum_{k=1}^{2}\frac{\text{count(k)}}{n} \quad (2)$$

Fig. 5: Storage representation of text format and proposed format

In other cases, the selected protein database contains the combination of all amino acids of 3/4/5/6 bit representation. Here, the size of the deduced protein sequence database file is given in Eq. 2. In the above cases, the size of the deduced protein database file is less than 7n which is required for any existing protein database. To evaluate our method we have collected few amino acid sequences of major organisms like Sheep, Lambda phage, E.coli, Chlamydomonas, Tetrahymena, Budding yeast, Fission yeast, Neurospora, Maize, Arabidopsis, Medicago truncatula, C.elegans, Drosophila, Xenopus, Zebrafish, Rat and Mouse from NCBI and reduced using proposed method. The result of this reduced space is shown in Table 4 and it found that the average reduced storage space is 43.77%, the minimum and maximum are 43.30%, 72.86%.

Table 4: Reduced storage space analysis on amino acid sequences

Name of organisms	Accession No. of Protein	Actual size (byte)	Final size (byte)	Reduced space (%)
Sheep	ABS70710.1	157	86	45.22
	CAF18242.1	172	96	44.19
	AAP86607.1	402	219	45.52
	AAP86606	402	219	45.52
	BAC10992	108	60	44.44
	BAC10991	315	168	46.67
	BAC10990	89	50	43.82
Lambda phage	CAA47810	554	304	45.13
	CAA70589	614	337	45.11
	CAA57857	1544	851	44.88
	CAA39202	562	308	45.20
	CAA58041	2104	1193	43.30
E-Coli	AAA23097	936	517	44.76
Chlamydo-	1Q90_A	292	162	44.52
monas	XP_001016824	2104	1004	52.28
Tetrahyme-na	XP_001008697.1	1429	796	44.30
	EAR88452.1	1429	796	44.30
	XP_001010932.1	1094	607	44.52
Budding yeast	Q23405.2	842	463	45.01
	O74653.1	840	454	45.95
	P18296.3	1828	1009	44.80
	Q9URV2.1	1458	799	45.20
	Q12381.1	906	500	44.81
Fission yeast	BAA21405.1	1241	685	44.80
	BAA21390.1	517	284	45.07
	BAA21388.1	2233	1235	44.69
	CAG99842.1	2233	606	72.86
Maize	1JQO	970	529	45.46
	AAN41252.1	1055	583	44.74
Neurospora	T51069	929	513	44.78
	AAK31733.1	1638	894	45.42
	XP_960681.1	1401	743	46.97
	XP_960662.1	1117	607	45.66
	ABZ49521.1	772	423	45.21
Maize	AAN41252.1	1055	583	44.74
	ABE98698.1	488	265	45.70
	AAN06977.1	351	195	44.44
	AAG41777.1	496	275	44.56
Arabidopsis	EAS41518.1	908	403	55.62
	EAZ09680.1	667	368	44.83
	EAZ09666.1	1058	584	44.80
Medicago	AAW78863.1	932	517	44.53
truncatula	NP_849731.1	975	533	45.33
	NP_174333.2	975	533	45.33
	NP_849727.1	800	441	44.88
	NP_850386.1	895	490	45.25
C.Elegans	CAM84804.1	3095	1672	45.98
	CAM84692.1	3095	1672	45.98
	CAM84693.1	2892	1559	46.09
	CAO82020.1	2371	1278	46.10
Drosophila	3BVX	1045	579	44.59
	3BVT	1045	579	44.59
	EAL30261.2	1448	796	45.03
	EAL30259.2	2171	1178	45.74
	EAL30255.2	1204	625	48.09
	EAL30252.2	1232	673	45.37
Xenopus	NP_989134.1	458	252	44.98
	NP_001085199.1	806	444	44.91
	NP_001085699.1	1022	562	45.01
	NP_001084574.1	1001	559	44.16
	CAJ82090.1	571	309	45.88
	CAJ82079.1	468	259	44.66
Zebrafish	2Z6G	780	428	45.13
	3B98	475	262	44.84
	BAF44666.1	2515	1387	44.85
	AAH91786.1	784	428	45.41
	AAH93193.1	422	234	44.55
Rat	1MAB	510	275	46.08
	XP_578142.2	519	288	44.51
	XP_215763.4	1705	916	46.28
	XP_575155.2	7613	4147	45.53
	XP_221512.4	535	285	46.73
	EDL81283.1	773	407	47.35
Mouse	AAI67479.1	517	278	46.23
	AAI36285.1	2137	1188	44.41
	AAI36286.1	2137	1188	44.41
	AAI56438.1	1500	820	45.33
Average	45.77			

The first screen of Fig. 5 contains the representation of Amylase in an ASCII format. When this amino acid sequence is stored using the proposed method, the storage has been nearly reduced to half and that is shown in the second screen of Fig. 5.

A large amount of biological data is semi-structured and stored in flat-files. These databases must be integrated with the structured data available in relational or object-oriented databases. The sequence matching process[8] is difficult, because string comparison takes more time. This method, converting the text to a binary format and then to bytes, reduces storage space and processing time. The probability of getting the deduced protein sequence lies between 42.86 and 87.17%. Further, this method provides a way to represent the data in a signal form and compression can be done using compression techniques.

REFERENCES

1. Michael Y. Galperin, 2007. The molecular biology database collection. Nucleic Acids Res., 35: 3-4. http://www.ncbi.nlm.nih.gov/pubmed/17148484.
2. Bairoch, A. and R. Apweiler, 1999. The SWISS-PROT protein sequence databank and its supplement TrEMBL. Nucleic Acids Res., 27:49-54. http://nar.oxfordjournals.org/cgi/content/full/27/1/49
3. Zoé Lacroix, 2002. Biological data integration: Wrapping data and tools. IEEE Trans. Inform. Technol. Biomed., 6: 123-128. Doi: 10.1109/TITB.2002.1006299
4. Andreas Teufel, Markus Krupp, Arndt Weinmann and Peter R. Galle, 2006. Current bioinformatics tools in genomic biomedical research. Int. J. Molecular Med., ISSN 1107-3756 17: 967-973 http://cat.inist.fr/?aModele=afficheN&cpsidt=17837351
5. Orpita Bosu and Simmender Kaur Thukral, 2007. Bioinformatics Databases, Tools and Algorithms. 1st Edn., Oxford University Press, ISBN: 9780195676839.
6. Lesk, A.M., 2001. Introduction to protein architecture: The structural biology of proteins. 1st Edn., Oxford University Press, ISBN: 0198504748
7. Branden, C.I. and J. Tooze, 2001. Introduction to Protein Structure., 2nd Edn., Garland Publishing, New York. ISBN: 13:9780815323051 ISBN: 10 0815323050

8. David R. Musser and Gor V. Nishanov, 2002. A fast generic sequence matching algorithm, dissertation submitted, Rensselaer Polytechnical Institute, New York. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.15.2586 http://www.literateprogramming.com/gsearch.pdf

9 Iskandar Ishak, Naomie Salim, 2006. Database Integration Approaches for Heterogeneous Biological Data Sources: An overview, Proceedings of the Postgraduate Annual Research Seminar, 202-206 http://66.102.1.104/scholar?hl=en&lr=&q=cache:LzN5nkpICEsJ:eprints.utm.my/3342/+Iskandar+Ishak,+Naomie+Salim,+2006