Article can be accessed online at http://www.publishingindia.com

Design and Development of Novel Methods for Searching Sequences

Dr.S. Vijayarani* Ms.S.Deepa**

Abstract

The concept of sequence data mining aimed to retrieve the frequent patterns in the sequences of products purchased by the customers through the time ordered transactions. Later, the application of sequence mining was extended to complex applications like telecommunication, network detection, DNA and protein sequence research. The technique of searching among sequence data is very important in many applications. A search technique is a technique for finding an item with specified properties among a collection of items. The searching process in sequence databases plays an important role in many application domains, mainly for information retrieval and data mining. When there are a number of stored objects, it will be too slow to linearly search all the stored items to find those that satisfy the query criteria. Hence various techniques and data structures are required to organise and manage the search process so that objects relevant to the query can be located quickly. In this research work, a new sequence search technique SSPP is proposed for performing sequence search operation in a retail dataset. The performance of SSPP technique is analysed to show the efficiency of the proposed technique when compared to other sequence search methods.

Keywords: Sequence, Sequence Database, Prefix Span, SSP, SSPP

1. Introduction

Data mining can also be described as data processing using sophisticated data search capabilities and statistical algorithms to discover patterns and correlations in large pre-existing databases. From these patterns, new and important information can be obtained that will lead to the discovery of new meanings which can then be translated into enhancements in many current fields. Sequences are an important kind of data which occur frequently in many fields such as medical, business, financial, customer behaviour, educations, security, and other applications (Esmaeili & Gabor, 2010). Sequential pattern mining is aiming at finding the frequently occurred sequences to describe the data or predict future data or mining periodical patterns (Agrawal & Srikant 1994, 1995). New algorithms and techniques (Dreyer *et al*., 1995; Seshadri *et al*., 1996) have also emerged recently to meet the new requirements of sequence data management.

Sequential pattern mining finds interesting sequential patterns among the large database. It finds out frequent subsequences as patterns from a sequence database. Let X = $\{i_1 . . . i_n\}$ be a set of items, each being associated with a possible set of attributes. The value of an attribute A of item i is denoted by i.A. An itemset consists of a non-empty subset of items. A sequence $\alpha = <A_1 \cdot \cdot \cdot A_n>$ is an ordered list of itemsets. An itemset A_i $(1 \leq i \leq l)$ in a sequence is called a transaction.

The length of a sequence is denoted by the number of transactions that are present in a sequence. A sequence $\alpha = <A_1 . . . A_n>$ is called a subsequence of another sequence $\beta = <B_1 . . . B_m>$ $(n \leq m)$, and β a super-sequence of α, if there exist integers $1 \leq i_1 < . . < i_n \leq m$ such that $X_1 Y_{i1}$, . . . , $X_n Y_{in}$. A sequential database is a set of 2-tuples (s_{id}, β), where s_{id} is a sequence-id and β is the sequence. A tuple (s_{id}, β) in a sequence database is said to contain a sequence λ if λ is a subsequence of β. The number of tuples in a sequence database containing sequence λ is called the support of λ, denoted by sup (λ). Given a positive integer minimum_sup as the support threshold, a sequence λ is considered to be a sequential pattern in sequence database Sdb if sup (λ) ≥ minimum_sup. The

* Research Scholar, Department of Computer Science, Bharathiar University, Coimbatore, Tamil Nadu, India

sequential pattern mining problem is used to find the complete set of sequential patterns with respect to a given sequence database and a support threshold minimum_ support (Agrawal & Srikant 1994, 1995).

Searching a list for a particular item is a general task. For performing search operations in data structures, there exist several different algorithms. There are many search algorithms in data structures that are used to search for a particular data item in a large amount of data. But still new algorithms yet to be developed to search for a particular sequence of data items in large volumes of data.

For example, in order to perform search of a particular item in a retail database, several algorithms such as linear search, binary search, fibonacci search etc are already available. These algorithms are used for finding whether the item (single), for example, bread is present in a particular transaction or not. But it is difficult to find more number of items, namely bread, butter and jam had been bought together in a particular transaction. In retail dataset, each individual item is stored with a unique product id. Hence the concept of integer sequence search can be used to find the sequence relationships among the items that were bought together. The sequential search algorithms perform search from the first data points to the end of the data sequence $s_1, s_2 : : : s_n$.

The rest of the paper is organised as follows: Section 2 describes the related work. Section 3 describes the problem objective and the proposed technique is explained in Section 4. Section 5 deals with the performance evaluation and the conclusion for the proposed technique is given in Section 6. References are given in Section 7.

2. Related Works

Searching an item or data from a large data set is a challenging task. Many numbers of searching algorithms are used for performing searching process. Some of the popular searching algorithms are Binary Search, Linear Search, Depth First Search, Breadth First Search, Binary Search Tree, Particle Swarm Optimisation, Genetic algorithm, etc. The simplest method of searching of an element is linear search. It is the simplest searching method which checks for an item one by one linearly (Beck, 1964; Bellman, 1963; Booch, 1995; Boyer & Moore, 1977).

In binary search technique, the divide and conquer strategy is used to find the search element. It divides the entire array of items in to two parts and verifies from the middle of the elements in the list. If the key value if less than the middle value then it searches to the left side till 0 and this process continues till N if the key value is greater than the middle value (Josuttis, 1999).

A number of algorithms have been proposed for performing searching operations for a sequence of strings in a large database. The Enhanced Checking and Skipping Algorithm (ECSA) is produced by the enhancement of the classical string searching algorithms by converting the character-comparison into character-access by using the condition type character access rather than the number-comparison and by starting the comparison at the latest mismatch in the previous checking which in turn increases the probability of finding the mismatch faster if there is any (Mhashi & Alwakeel, 2010).

The Boyer-Moore algorithm performs the search from right to left in the pattern. The algorithm places the pattern over the leftmost characters in the text and attempts to match it from right to left. If there is no mismatch, then the pattern has been found. Or else the algorithm performs a shift, which is an amount by which the pattern is moved to the right before a new matching attempt is undertaken (Boyer & Moore, 1977).

In KMP algorithm, each time when a mismatch is found, the false start consists of characters that were already examined. This avoids the repetitive comparisons with the known characters. The algorithm can be arranged so that the pointer in the text is never decremented. To accomplish this, the pattern is preprocessed to obtain a table that gives the next position in the pattern to be processed after a mismatch (Knuth & Pratt, 1977).

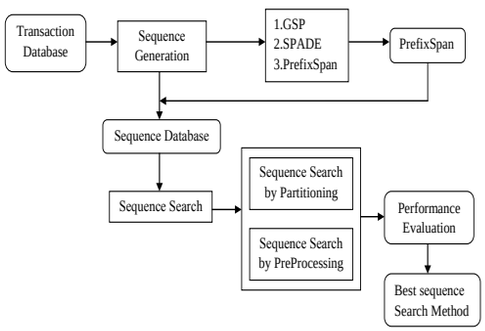
When the alphabet size is large and the length of the pattern is small, it is not good to use Boyer-Moore's bad-character technique. Instead of this method, the bad-character shift of the right-most character of the window to compute the value of the shift is found. These values for the shift are calculated in the preprocessing stage for all the characters in the alphabet set. Hence, the Horspool algorithm is more efficient in practical situations where the alphabet size is large and the length of the pattern is small (Horspool, 1980).

3. Problem Objective

The sequential pattern mining aims to retrieve the frequent sequences in the given sequence database based on the user defined minimum support. From the datasets, using several sequence pattern mining algorithms, we can generate the sequences and these generated sequences are stored in a sequence database. The main aim of the proposed search technique is to perform the search operation in the sequence database and as well to count the occurrences of the search sequence.

In this paper, a new technique, Sequence Search by Preprocessing (SSPP) is proposed for searching a sequence in s sequence database. Initially, the sequences are generated from the data set by using several sequence generation algorithms. Important sequence generation algorithms are Generalised Sequential Patterns (GSP), Sequential Pattern Discovery using Equivalent classes (SPADE), Prefix-Projected Sequential Pattern Growth (PrefixSpan), Frequent Pattern-Projected Sequential Pattern Mining (FreeSpan), Sequential PAttern Mining (SPAM), Incremental Sequence Extraction (ISE), etc. In this research work, three sequence generation algorithms, namely GSP, SPADE, and PrefixSpan algorithms are used for generating sequences. By measuring the efficiency of these algorithms, the PrefixSpan algorithm performance is better than GSP and SPADE (Vijayarani & Deepa, 2013). The sequences generated by the Prefixspan algorithm are stored in a sequence database. In this research work, a new search technique is proposed to perform the search process and to find the number of occurrences of a particular sequence in a sequence database. The techniques proposed for performing search operations is Sequence Search by PreProcessing (SSPP) and its performance is compared with another sequence search method, that is, Sequence Search by Partitioning.

Figure 1: System Architecture for Sequence Search



3.1. Dataset

The dataset used in this paper is taken from Frequent ItemSet Mining Repository (http://fimi.ua.ac.be/data/retail.dat). Retail dataset is used in this research work. It is a real time dataset collected from a Belgian Retail Supermarket store. The dataset consists of 88,163 transactions and 16,440 different products that are sold in various transactions carried over in a certain period of time. The transactions consist of unique ids that are given for each product that was provided by the store.

Table 1: Sample dataset

Transaction	*Sequence of Products*
T1	36 37 38 39 40 41 42 43 44 45 46
T2	38 39 47 48
T3	38 39 48 49 50 51 52 53 54 55 56 57 58
T4	32 41 59 60 61 62
T5	36 37 38 39
T6	50 51 52 53
T7	52 53 54 55 56 57 58

3.2. The Prefixspan Algorithm

The PrefixSpan algorithm (Jian *et al*., 2004) is a well-known pattern-growth approach. The major idea of PrefixSpan algorithm is that any frequent subsequences can always be found by growing frequent prefixes. It divides database into smaller projected databases and solves them recursively. It examines only the prefix subsequences and projects only their corresponding postfix subsequences into projected databases. In each projected databases, sequential patterns are grown by exploring only local frequent patterns. Since no candidate sequence needs to be generated, the database need not be scanned multiple times. Since Prefix-projection substantially reduces the size of projected databases, it leads to efficient mining of sequential patterns. The PrefixSpan algorithm is applied to generate the sequential patterns from the retail database.

The PrefixSpan Algorithm

PrefixSpan(α, i, S| α)

Begin

1. Scan S|α once, find the set of frequent items b such that

- b can be assembled to the last element of α to form a sequential pattern; or
- <b> can be appended to α to form a sequential pattern.

2. For each frequent item b, appended it to α to form a sequential pattern α' and the output α';

3. For each α', construct α'-projected database S| α' and call PrefixSpan(α', i+1,S| α').

End

Table 2: Sample sequences produced from Retail dataset by PrefixSpan algorithm

S. No	SEQUENCES PRODUCED
1	38
2	39
3	53
4	38, 39
5	38, 53
6	39, 53
7	38, 39, 53

3. Search Techniques

3.1. Sequence Search by Partitioning

This method consists of two steps namely partitioning and searching. In partitioning, the sequence database Sdb is partitioned into different tables as $T_1, T_2 \ldots T_n$ based on the length of the sequences. For example, the sequence length is one, two, three, etc. Sequences with one item are stored in table T_1 and the sequences with two items are stored in another table T_2 and so on. In order to perform search process, the input search sequence Ss is required. Next, the input search sequence length (L) is calculated. Based on the search sequence length (L), the search starts from the table T_L and continues up to the table T_n. The search Sequences Ss are retrieved and the count value is calculated.

The SSP Algorithm:

Input: Sequence database Sdb, Search sequence Ss.

Output: (i) Search successful or unsuccessful (ii) Count.

Method:

1. Consider the input sequence database, Sdb=<$s_1,s_2\ldots..$ $s_{n>}$, $s_j \in I$, where j=1 to n be the set of sequences and I=<$i_1,i_2\ldots.i_m$> where i=1 to m be the set of items.

2. Initialize Count=0 and L=0.

3. Partitioning:

3.1 Partition Sdb into T tables based on the sequence length.

3.2 Consider the search sequence Ss and calculate its length (L).

4. Searching:

4.1 Start search from T_L to T_n.

4.2 If (Ss $\in T_L$) then display the search sequence Ss;
Increment the value of count and L;

4.3 If (L>n) Then display the value of count;
Else Goto step 4.2;

4.4 Else Increment the value of L and Goto step 4.2;
End

3.2. Sequence Search by Pre Processing

This method, too, involves two steps namely preprocessing and searching. In preprocessing, all the combinations of the items $c_1,c_2\ldots c_q$ in sequence database Sdb are produced and their count of occurrence in sequence database is calculated and stored in a combinations table C. To perform searching, an input search sequence Ss is given. The search directly goes to the combinations table C where the entire set of combinations of items is stored and the particular search sequence Ss is checked in C. If the search sequence Ss is present, their count that is already stored in the combinations table C is retrieved and if the sequence Ss is not present, the sequence not present message will be displayed.

This method involves initial finding of combinations of items and their count of occurrence in the database. Since the combinations and their count are found out in advance, during the search sequence, the search directly goes to combinations table and the search sequence and the count can be easily retrieved.

The SSPP Algorithm:

Input: Sequence database Sdb, Search sequence Ss.

Output: i) Search successful or unsuccessful ii) Count.

Method:

1. Consider the input sequence database, Sdb=<s_1,s_2…..$s_{n>}$, s_i ∈ I, where I=<i_1,i_2….i_m> be the set of items
2. Preprocessing:
   2.1 Split Sdb into individual items as J=j_1, j_2,…..j_p where J ⊆ I.
   2.2 Generate all the possible combinations of each individual item in J and store them in the combinations table C and count their occurrences.
3. Searching:
   3.1 Consider the input search sequence Ss.
   3.2 If (Ss ∈ C) then display the sequences and their Count;
   Else
   Process Terminated.

4. Performance Evaluation

The concept of efficiency (or complexity) is important when comparing algorithms. For performing the task of searching in large amount of data, the choice among alternative algorithms becomes important because they may differ in efficiency. One way to compare algorithms is to compare the performance of the algorithms in terms of how quickly they solve the problem. Another way of comparing algorithms is to look at the amount of space (memory) they require to perform the search operation.

To test the proposed method, a series of performance studies were conducted. A performance test is focused on the memory used and in addition, the execution time of the two methods is also analysed. The evaluation was performed on PC Intel Pentium processor, 2GB RAM, OS Windows 7 Ultimate 32-bit. The subsequent tests compare performance of two different search techniques on retail dataset. The performance of these two search techniques are analyzed under various criteria such as various dataset sizes and various search sequences length. The different sizes of dataset used in this work are 250, 1000 and 2000. The different search sequence lengths are 2, 6 and 10.

The results for the search sequence (38, 39) and its occurrence count are shown in Table 3.

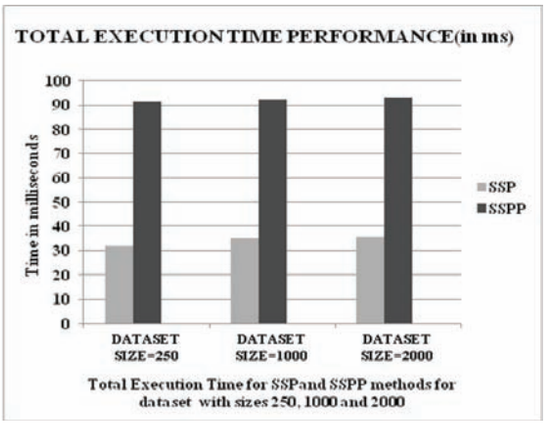
Table 3: Sample output for search sequence

Search Sequence	*Output*	*Count*
38, 39	[38, 39] [38, 39, 53]	2

Table 4: Total Execution Time of SSP, SSI and SSPP techniques for datasets of various sizes

Algorithm	*Dataset size*	*Total Execution Time(in ms)*
SSP	250	32.0
	1000	35.2
	2000	35.9
SSPP	250	91.2
	1000	92.3
	2000	92.9

Figure 2: Total Execution Time of SSP and SSPP techniques for datasets of various sizes

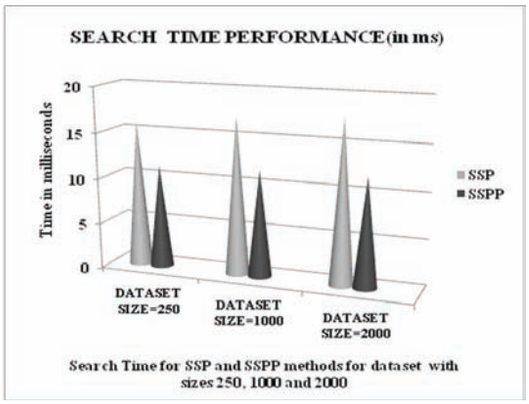


The graph shown in Figure 2 depicts the total execution time taken for searching the sequences by SSP and SSPP techniques. From the results, we observed that the SSP technique takes minimum execution time than SSPP technique.

Table 5: Search Time of SSP and SSPP techniques for datasets of various sizes

Algorithm	*Dataset size*	*Search Time (in ms)*
SSP	250	16.0
	1000	17.3
	2000	17.8
SSPP	250	11.3
	1000	11.7
	2000	11.9

Figure 3: Search Time of SSP and SSPP techniques for datasets of various sizes

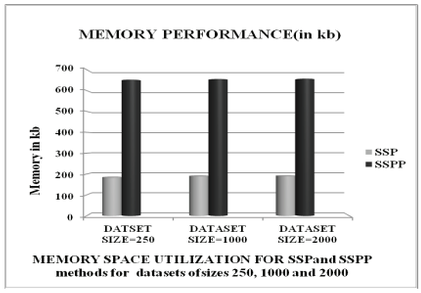


The graph shown in Figure 3 shows search time of two search techniques. The result shows that the search time of SSPP require minimum search time than SSP.

Table 6: Total Memory Space of SSP and SSPP techniques for datasets of various sizes

Algorithm	*Dataset size*	*Total Memory (in kb)*
SSP	250	184.0
	1000	190.0
	2000	190.8
SSPP	250	651.2
	1000	654.4
	2000	656.0

Figure 4: Total Memory Space of SSP and SSPP techniques for datasets of various sizes

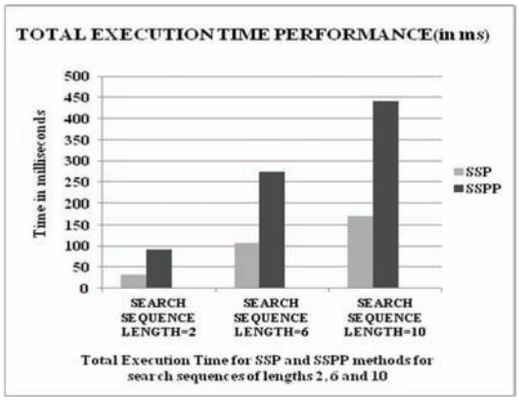


The graph in Figure 4 shows the total memory space utilised for searching the sequences by SSP and SSPP techniques. From the results, we observed that the SSP technique occupies minimum memory space than SSPP technique.

Table 7: Total Execution Time of SSP and SSPP methods for search sequences of various lengths

Algorithm	*Sequence Length*	*Total Execution Time(in ms)*
SSP	2	32.0
	6	106.8
	10	170.0
SSPP	2	91.2
	6	273.6
	10	440.0

Figure 5: Total Execution Time of SSP and SSPP methods for search sequences of various lengths



The graph in Figure 5 shows total execution time for searching the sequences by SSP and SSPP techniques. The results showed that the SSP occupies minimum execution time.

Table 8: Search Time of SSP and SSPP techniques for search sequences of various lengths

Algorithm	*Sequence Length*	*SearchTime (in ms)*
SSP	2	16.0
	6	53.4
	10	85.0
SSPP	2	11.4
	6	34.9
	10	55.0

The graph shown in Figure 6 shows the search time for searching the sequences by SSP and SSPP techniques. The results show that the SSPP occupies minimum search time than SSP.

Figure 6: Search Time of SSP and SSPP techniques for search sequences of various lengths

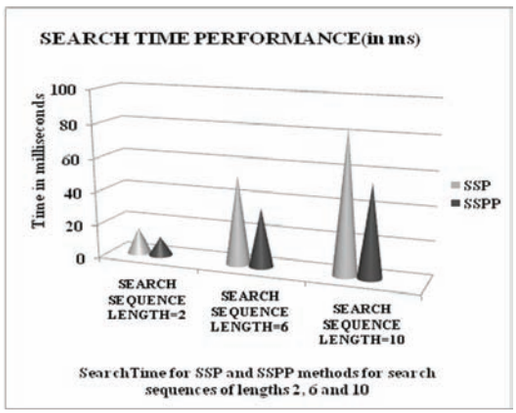
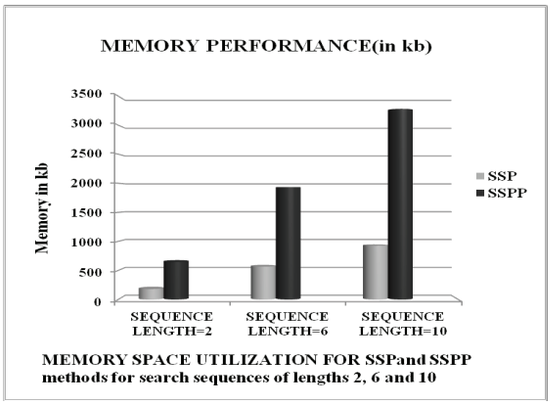


Table 9: Total Memory Space of SSP and SSPP techniques for search sequences of various lengths

Algorithm	*Sequence Length*	*Total Memory (in kb)*
SSP	2	182.0
	6	567.6
	10	928.0
SSPP	2	651.2
	6	1934.4
	10	3280.0

Figure 7: Total memory space of SSP and SSPP techniques for search sequences of various lengths



The graph in Figure 7 shows the total memory space utilised for searching the sequences by SSP and SSPP techniques. From the results, we observed that the SSP technique occupies minimum memory space than SSPP technique.

5. Conclusion

Nowadays many applications involve the management of sequence data. However, traditional relational database techniques are insufficient in handling queries on sequence databases. Therefore sequence data searching algorithms are in demand. Advanced searching algorithms are likely to be useful in practical applications and they present a number of interesting problems in the analysis of algorithms. There are a number of algorithms for searching a single integer value or for searching a sequence of strings. This research work analyzed the concept of searching a sequence of integers and a new search technique SSPP is proposed for performing search process for a sequence of integers on retail dataset. By analysing the experimental results, we come to know that the SSP technique needs minimum execution time and SSPP require minimum search time for searching the sequence. In terms of memory utilisation, SSP occupies less amount of memory when compared with SSPP technique.

References

Agrawal, R., & Srikant R., (1994). *Fast Algorithms for Mining Association Rules.* 20th International Conference on Very Large Data Bases (pp. 487-499).

Agrawal R., & Srikant, R., (1995). *Mining Sequential Patterns.* 11th International Conference on Data Engineering. IEEE Computer Society Press, Taiwan (pp. 3-14).

Beck, A.(1964). *On the linear search problem. Israel Journal of Mathematics*, 2(4), 221-228.

Bellman, R. (1963). *An optimal search problem*, *SIAM Review*, 6, 2, 168-174.

Booch, G. (1995). *Object Oriented Analysis and Design* (2nded), Addison-Wesley.

Boyer, R. S. & Moore, J. S. (1977). *The Boyer-Moore Algorithm.*

Collins, W. J. (1992). *Data Structures. first Edition* Addison-Wesley publishing company, page 397, U.S.A

Knuth, D. & Pratt, V. (1977). *Knuth-Morris-Pratt Algorithm. SIAM Journal on Computing*, 6(1), 323-350.

Dreyer., W, Dittrich., A. K., & Schmidt., D. (1995). *Using the CALANDA Time Series Management Systems.* In Proceedings of the ACM SIGMOD Conference on Management of Data (pp. 489).

Horspool, R. N. (1980). Practical fast searching in strings. *Software- Practice Experience*, 10(6), 501-506.

Jian, P., Han, J., Mortazavi-Asl, B., Wang, J., Pinto, H., Chen, Q., Dayal, U., & Hsu, M. C. (2004). *Mining Sequential Patterns by Pattern-Growth: The Prefix-Span Approach*. IEEE Transactions on Knowledge and Data Engineering, 16(10).

Josuttis, N. M. (1999). *The C++ standard library: A tutorial and reference*. Addison-Wesley, Reading.

Esmaeili, M., & Gabor, F. (2010). Finding sequential patterns from large sequence data. International Journal of Computer Science Issues, 7(1), 1694-0814.

Mhashi, M. M., & Alwakeel, M. (2010). New enhanced exact string searching algorithm. *International Journal of Computer Science and Network Security*, 10(4), 13-20.

Seshadri, P, Livny, M. & Ramakrishnan, R. (1996). *The Design and Implementation of A Sequence Database System.* In Proceedings of 22nd VLDB Conference (pp. 99-110).

Yun, U. (2007). Analyzing Sequential Patterns in Retail Databases. *Journal of Computer Science and Technology*, 22(2), 287-296.

Vijayarani, S., & Deepa, S. (2013). An Efficient algorithm for sequence generation in Data mining. *International Journal of Cybernetics and Informatics*. 3(1), 21-30.

Li, Y., Lauria, M., & Bundschuh, R. (2004). *Using Hybrid Alignment for Iterative Sequence Database Searches.* Currency and Computation: Practice and Experience.

Zabinsky, Z. B., & Smith, R. L. (1990). *An adaptive Random search algorithm with linear complexity in dimension*. Technical Report 90-15.