|  |  |  |
| --- | --- | --- |
|  | Available online at www.sciencedirect.com |  |
| **ScienceDirect** |
| AASRI Procedia 4 ( 2013 ) 313 – 318 |
| 2013 AA ASRI Conf ference on In ntelligent S ystems and d Control  Const traint Ba ased Seq quential P Pattern M Mining i in Time Series  Database es - A tw wo Way A Approac ch  Van ngipuram R Radhakrish hnaa\*,Chint takindi Srin nivasb,Dr.C C.V.Guru R Raoc  *aDepartm ment of Informatio on Technology, V VNR Vignana Jyo othi Institute of En ngineering and Te Technology, Hyder rabad, INDIA*  *bA Associate Profess or , Department o of Computer Scie ence and Enginee ering, Kakatiya In nstitute of Techno ology and Science e, Warangal, IND DIA*  *cProfessor & He ead, Department of Computer Scie ence and Enginee ering, S. R Engine eering College,H Hasanparthy, War rangal, INDIA* | | |

**Abst tract**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Most t of the patter s search algorithm ms in literature e are mostly str ring based and | | | do not concen ntrate on finding g sequential pa atterns | |
| with | constraints in a a given databas se. Also in quer ry languages su ch as SQL or M MySQL, the sel ect clause does s not allow the u use of | | | |
| the n non-aggregate f functions as pa art of query com mpilation. The | | objective of th his work is to p propose a patte ern mining algo orithm | | |
| whic ch may be embe edded into SQL L or MySQL Q Query language es so that we ca an search for pr resence of sequ uential pattern | | | | in the |

datab base. The algori ithm makes use e of sliding sequ uential patterns s with three look k-ahead elemen nts in the event of any mismatc ch.

|  |
| --- |
| © 2013 The Authors. Published by Elsevier B.V.© 20 013 The Auth ors. Published d by Elsevier B B.V. Open access under [CC BY-NC-ND license.](http://creativecommons.org/licenses/by-nc-nd/3.0/)  Selection and/or peer review under responsibility of American Applied Science Research Institute earch Institute  ***Keyw words :*** sequence p pattern;look ahea ad; time series;seq quence tuple |
| **1. In ntroduction**  In n this paper th he idea is to fir rst perform pre eprocessing of f the sequentia al pattern befo ore the search process. Inste ead of |

searc ching for a st tring based pa attern, we per form search f for a sequence e of tuple or r record values with user de efined

cons straints. The el lements of pat ttern considere ed are numeric cal type.

Most of the al lgorithms desi igned in litera ature are strin ng based and are not devel loped by cons sidering seque ential

patte erns with cons straints. An att tempt is made e in this work towards this d direction by co onsidering a ti ime series data abase.

We search for a sequential pat ttern of intere est in a given n database wit th each patter rn element im mposed with a a user

defin ned constraint t.

|  |  |  |  |
| --- | --- | --- | --- |
| The | algorithm de eveloped may | be embedded d in to any e existing query | languages so o that the use e of non aggr regate |

func ctions is possib ble which mak kes the compl lexity of the q query get redu uced in terms o of query proce essing and als so the

|  |  |
| --- | --- |
| time e efficiency. S Section III, IV V describes th he proposed a algorithm wit th a working | example. The e objective of f this |
| algo rithm is to ma ake the pattern n shift by a len ngth more than n the pattern l length as comp mpared to the e existing algorit thms.  The efficiency can n be seen from m the moves m made by the pa attern itself.  \* Corresponding a author. Tel.: +970 00684242 | |

*E-mail address:* ra adhakrishna\_v@v vnrvjiet.in

2212-6716 © 2013 The Authors. Published by Elsevier B.V. Open access under [CC BY-NC-ND license.](http://creativecommons.org/licenses/by-nc-nd/3.0/)

Selection and/or peer review under responsibility of American Applied Science Research Institute

doi: 10.1016/j.aasri.2013.10.046



314  *Vangipuram Radhakrishna et al. / AASRI Procedia 4 ( 2013 ) 313 – 318*

**2. Related Works**

In [1] Wang and Kobayashi propose a pattern search algorithm for network security applications that computes a function called next before the start of the search process. The value defined by next is used in the event of a mismatch between the pattern and the text. In [8] the authors design a pattern search algorithm using two sliding windows. In [6] algorithm uses the concept of implications to find the relation between the pattern elements and is designed to find the sequence pattern in a given database. In [9] algorithm designed performs the two way pattern search using sliding patterns. The algorithm in [10] uses three sliding windows.

**3. Proposed Work**

The algorithm consists of two steps.

a) Pre-process sequence pattern with constraints.   
b) Search for sequence pattern in the given database.

***3.1 Preprocessing phase***

***3.1.1 Left Shift failure function***

1 ; pattern[length (pattern) -1]=L1

2 ; pattern[length (pattern) -1]pattern[length (pattern) -2]=L2L1

length (pattern) -index ; pattern[index]pattern [index+1] pattern [index+2] =L1L2L3

left\_shift [L1, L2, L3] = minimum length (pattern) +1 ; pattern [0] pattern [1] =L2L3

length (pattern) +2 ; pattern [0] =L3

length (pattern) +3 ; else *(1)*

***3.1.2 Right Shift failure function***

length (pattern) +2 ; pattern [length (pattern) -1] = L1

length (pattern) +1 ; pattern [length (pattern) -1] pattern [length (pattern) -2] = L2L1

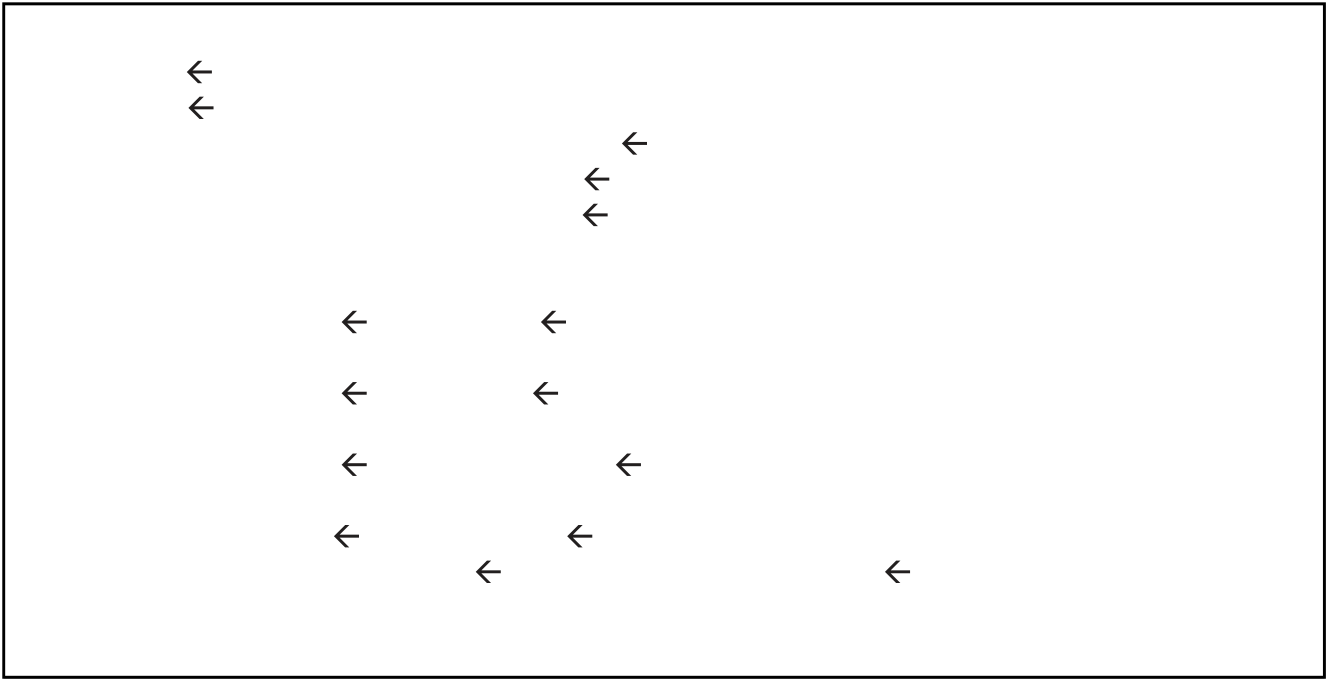
index +3 ; pattern [index] pattern [index+1] pattern [index+2] = L1L2L3

Right\_shift [L1, L2, L3] = minimum 1 ; pattern [0] pattern [1] =L2L3

2 ; pattern [0] =L3

length (pattern) +3 ; else *(2)*

In the preprocessing phase failure functions are first computed. In case of any mismatch in the search process the algorithm uses three attribute values of the input present immediately after the aligned sliding pattern. These attribute values are called look ahead elements.



*Vangipuram Radhakrishna et al. / AASRI Procedia 4 ( 2013 ) 313 – 318*  315

The algorithm for the preprocessing of the sequence pattern is given in fig 1 below.

Begin algorithm   
left\_shift length (pattern) +3;   
right\_shift length (pattern) +3;   
for each pattern element Pattern [index] from i 0 to length (pattern) -3 do Compute Next [index] as Next [index]   
 length (pattern) –index; next [index] index+2   
end for   
 If (pattern [length (pattern) -1] = =L1) then   
 {left\_shift 1; right\_shift length (pattern) +2}   
else If (pattern [length (pattern) -1] pattern [length (pattern) -2] = =L2L3) then {left\_shift 2; right\_shift length (pattern) +1}   
else If (pattern [index] pattern [index+1] pattern [index+2] = = L1L2L3) then {left\_shift next[i]; right\_shift next[i]}   
else if (pattern [0] pattern [1] = =L2L3) then

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| {left\_shift | m+1; right\_shift | | 1} | 2} |
| else if (pattern [0] ==L3) {left\_shift | | length (pattern) +2; right\_shift | |
| end if | | | |

end of algorithm

Fig. 1. Algorithm for computing shift-left and shift-right failure functions

***3.2. Search for Sequence Pattern in the Database***

This phase consists of searching for the sequence pattern over a specific attribute or column in given database. Initially the pattern is aligned with left end and right end of the input database. Algorithm scans concurrently from both sides of database to find the presence of user defined sequence pattern. Initially, algorithm starts by searching for presence of the sequence pattern from left end of the database.

While searching from left if a mismatch occurs, the sequence pattern window is shifted rightward by a shift value defined by the left shift failure function. The algorithm now performs the search process by scanning from the rightmost end of the database moving towards left and if a mismatch occurs; shift-right failure function is used to shift the pattern towards left. Algorithm terminates when the pattern is found or the left and right indices over cross each other.

**Step1:** At start the sequential pattern is aligned with left and right ends of the input database. To reduce the number of comparisons made, only the terminal elements of sequence pattern are compared. If the dead end elements match with pattern elements then we compare other pattern elements.

In an event of any mismatch, the algorithm moves to step2 otherwise the searching continues till the sequential pattern of interest is found. On success it reports the index position of the sequence pattern.

**Step2:** When a mismatch occurs from left end of the database while searching from the left, algorithm uses shift-left function to shift the sequence pattern making use of three attribute values of the input database which immediately appear after the aligned left sliding pattern. Similarly, shift value is computed using right failure function if a mismatch occurs from right end of database. This process continues until first occurrence of sequence pattern is found in the input database from either side or until both indices of sliding patterns over cross each other. The detailed algorithm for the same is given below in fig 2.

|  |  |
| --- | --- |
| 316 | *Vangipuram Radhakrishna et al. / AASRI Procedia 4 ( 2013 ) 313 – 318* |

|  |  |  |  |
| --- | --- | --- | --- |
| Left0; // index used from left Rightn-m; // index used from right found false; // initial value while (Left < = Right) do   leftindex length(pattern)-1;   rightindex length(pattern)-1;  if (Pattern [0] ==inputdB [Left])   while (leftindex > 0) do   if (Pattern[leftindex] ==inputdB [Left])   |  |  |  | | --- | --- | --- | | leftindex |  | leftindex-1; |   else   break;   end while  if (Pattern [0] ==inputdB[Right])   while(rightindex>0) do   if (P[rightindex] ==T[rightindex+Right])   rightindex rightindex-1; else   break;   end while  if (leftindex = = 0) {sequence retrieved left end at: Left} ; exit from outer loop; if(rightindex = = 0) {sequence retrieved at right end at: Right}; exit from outer loop; Left  Left + shiftL (text substring (Left+m, Left+m+3)); RightRight-shiftR (text substring (Right-3, Right));  if (Left > Right) {search is failure} exit from outer loop; |

Fig. 2. Searching Algorithm

**4. Working Example**

|  |  |  |  |
| --- | --- | --- | --- |
| **P1** | **P2** | **P3** | **P4** |

where P1 = tuple.value < tuple.previous.value

P2 = tuple.value < tuple.previous.value ^ 40 < tuple.value < 50

P3 = tuple.value > tuple.previous.value ^ tuple.value < 52

P4 = tuple.value > tuple.previous.value.

and the operator ‘^’ indicates logical AND operation.



*Vangipuram Radhakrishna et al. / AASRI Procedia 4 ( 2013 ) 313 – 318*  317

Think of a scenario as in [8] where we are interested to find trends in the temperature for four consecutive days, i.e every possible instance where a sequence pattern consists of two immediate falls followed by two immediate hikes and the drops are such that the temperature to lie between 40oc and 50oc.

Also the first increase does not let the temperature move beyond 52oc.Here, we need to retrieve the sequence of Quintuple (A, B, C, D, E) with B.value < A.value, C.value < B.value, 40 <C.value<50, C.value<D.value, D.value<52, D.value<E.value holding well. The optimal set of constraints called as pattern elements are thus derived as shown above. The working of the algorithm is as explained below

**First iteration:** The search process begins by comparing pattern element at index=0 to the element of the inputdB at index=0. As there is a mismatch and also as all other cases violate, So left\_shift failure function computes shift value as length(pattern)+ 3 = 7. The algorithm shifts sequence pattern to the right by 7 units. At the end of first iteration the algorithm makes the pattern indices from 0 to 3 aligned to indices of inputdB from 8 to 12.

**Second iteration:** Now thesequence pattern aligned at right end of the input database is considered for search process, pattern element at index 0 is compared with the database element at index 10. Since there is a mismatch, the immediate three attribute values to the left of the input database at indices 7, 8, 9 are considered to compute shift value using right\_shift failure function. We stop the search process as there is a mismatch and start the search process from left end.

**Third iteration:** The elements of the sequence pattern from indices 0 to 3 are compared with the input database attribute values from indices 8 to 11 and it is a success. So stop and output the sequential pattern.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **10** | **40** | **70** | **80** | **65** | **42** | **53** | **48** | **46** | **49** | **60** | **75** | **80** | **60** |

**P0 P1 P2 P3 from left end first iteration**

**from right end second iteration P0 P1 P2 P3**

**from left end third iteration P0 P1 P2 P3 [Best case Move]**

Fig. 3. Working Example

**5. Conclusion**

A sequential pattern mining algorithm with the pattern elements consisting of the user defined constraints is presented in this work. Failure functions are designed to handle the case of mismatch. The process of concurrent searching from the both ends of the database makes algorithm more efficient for the worst case situation when required sequence pattern is even at the end of the input. The reduction in the number comparisons done is visible from the best case move where algorithm shifts the pattern by a shift value greater than length of pattern when compared to algorithm used in [6].

318  *Vangipuram Radhakrishna et al. / AASRI Procedia 4 ( 2013 ) 313 – 318*

**References**

[1] Knuth, D.E., J.H. Morris and V.R. Pratt. Fast pattern matching in strings. SIAM Journal of Computing. 6(2).1977, 323-350.

[2] Boyer, R. S. AND Moore, J. S. A fast string searching algorithm. Communications of the ACM. 20(10).1977, 762–772.

[3] R. Nigel Horspool. Practical fast searching in strings. Software Practice and Experience. John Wiley and Sons. (10) 1980, 501-506.

[4] T.BERRY AND S.RAVINDRAN A Fast String Matching Algorithm and Experimental Results, Proceedings of Prague Stringology   
 Club, workshop’99   
[5] Handbook of Exact String Matching Algorithms.

[6] Expressing and Optimizing Sequence Queries in Database Systems.

http://www.cs.ucla.edu/~zaniolo/papers/todsjune04.pdf   
[7] Wang, Y. and H. Kobayashi. High performance pattern matching algorithm for network security.6 (10). 2006, 83-87.

[8] Mjad Hudaib, Rola Al-Khalid, Dima Suleiman, Mariam Itriq and Aseel Al-Anani. A Fast Pattern Matching Algorithm Using Two   
 Sliding Windows, Journal of computer science. 4(5) 2008, 393-401.

[9] V.Radhakrishna, B.Phaneendra, V.Sangeeth Kumar. A Two Way Pattern Matching Algorithm Using Sliding Patterns. In the Proceedings of 3rd IEEE International Conference on Advanced Computer Theory and Engineering. (2) 2010, 666-670.

[10] V.Radhakrishna, C.Srinivas, Dr.C.V.Guru Rao. High Performance Pattern Search algorithm using three sliding windows. IJCET 3(2) 2012, 543-552. Journal Impact Factor: 3.9580 calculated by GISI.