METISP Algorithm

1. Input the sequential database *D*, minimum support threshold (*min\_sup*), minimum gap (*mingap*), maximum gap (*maxgap*), sliding window (*swin*) and duration (*dun*) are intergers.

DB contains sequences

S : {t1{b}t2{a c}t3{a, b, c}}

vector<int> items contain int values as a,b,c ,

timeOcc is int value as t1,t2,t3.

Transaction T contains items, timeOcc as t2{a c}

Sequential seq contains Transaction

DB contains n Sequential

1. Scan *DB* to determine the set of frequent items.

int \* frequency contain support of each item, that is, the number of Sequential contains item.

1. Construct *x-Tidx*:

Delete infrequent items

Sau bước xóa những item có frequency không thỏa mãn minsup, còn lại những item thỏa mãn (xóa cả tần số tương ứng trong frequency).

Tạo pattern từ item thỏa mãn

set *p* = [{*x*}];

Xây dựng là time-idex cho pattern p như sau:

*p*-Tidx= (it1, lst1, let1, it2, lst2, let2,…, it*l*, lst*l*, let*l*), where it*i*, lst*i* and let*i* (*i=* 1 to *l*) are the initial time, last-starting time and last-ending time of the *i*th repetition of *p* in *ds*

|  |  |
| --- | --- |
| SID | Sequences |
| 1 | {3{*c*}, 5{*a*, *f*}, 18{*b*}, 31{*a*}, 45{*f*}} |
| 2 | {6{*a*, *c*}, 10{*b*}, 17{*e*}, 18{*a*}, 24{*c***,** *d*}} |
| 3 | {1{*b*}, 20{*b*, *g*}, 27{*e*}, 34{*d*, *g*}, 35{*g*}} |
| 4 | {5{*a*}, 10{*d*}, 21{*c*, *d*}, 26{*e*}} |

*Ví dụ: pattern {a} nằm trong chuỗi SID 1,2,4. Ta tính* *{a}-Tidx tương ứng với chuỗi 1,2,4.*

*Với chuỗi 1, có {a} xuất hiện 2 lần, {a}-Tidx1= {5,5,5,31,31,31}*

*Giả sử pattern loại 1**{{a}{b}} nằm trong chuỗi 1,2. Ta tính {a}-Tidx tương ứng với chuỗi 1,2.*

*Với chuỗi 1, có {{a}{b}} xuất hiện 2 lần, {a}{b}-Tidx1= {5,18,18,5,31,31}*

*Giả sử pattern loại 1{{a,c}} nằm trong chuỗi 1,2,4. Ta tính {a,c}-Tidx tương ứng với chuỗi 1,2,4.*

*Với chuỗi 1, có {{a,c}} xuất hiện 2 lần, {a,c}-Tidx1= {3,3,5,3,3,31}*

(3.2)

construct *p*-Tidx, where *p*-Tidx is the time index of *p*;  is represented by *.*

Output: p-Tidx

(3.3) input : P-Tidx

Output: VTP

(3.4) input: VTP

Output: Transaction => items

*(3.5) input: items*

*Out put: patterns thỏa mãn.*

Let the time -index of *p* in *ds* be denoted as (it1, lst1, let1, it2, lst2, let2,…, it*l*, lst*l*, let*l*).

The VTP (valid time period) to generate type-1 pattern will satisfy the following condition: (let*j*+ *mingap*) ≤ VTP1 ≤ minimum of (lst*j* + *maxgap*, it*j*+*dun*) for *j* = 1 to *l*, where *dun* is the duration.

From VTP1 choose valid transaction whose items can create type-1 pattern, are transaction having occurring time belong to VTP1

For the type-2 pattern, the corresponding VTP2 should satisfy:

(let*j* – *swin*) ≤ VTP2 ≤ minimum of (lst*j* + *swin*, it*j*+*dun*) for *j* = 1 to *l*

From VTP choose valid transaction whose items can create type-2 pattern, are transaction having occurring time belong to VTP 2

(3.4) For each item *x* found in VTPs of type-1 pattern with support ≥ *min\_sup*×|*D*|:

* 1. Form type-1 pattern *p*’ by adding {*x*} after the last element of *p* and set *p* = *p*’.
  2. Scan *D* to construct *p*-Tidx.
  3. Recursively until no sequential pattern can be found.

(3.5) For each item *x* found in VTPs of type-2 pattern with support ≥ *min\_sup*×|*D*|:

1. Form type-2 pattern *p*’ by setting the last element of *p*’ as the union of {*x*} and the last element of *p*. Set *p* = *p*’..
2. Scan *D* to construct *p*-Tidx.
3. Recursively until no sequential pattern can be found.

. (4) Return *p* and *p*-Tidx.