

An efficient, versatile and scalable pattern growth approach to mine frequent patterns in unaligned protein sequences

Kai Ye¹, Walter A. Kusters², Adriaan P. IJzerman¹

¹Division of Medicinal Chemistry, LACDR, Leiden University, Leiden, The Netherlands

²Leiden Institute of Advanced Computer Science, Leiden University, Leiden, The Netherlands.

A *pattern* is an ordered series of *items*, where each item is either one of 20 residues or 3 special *wildcard* characters, x , $x(i,j)$ or $*$ in which x matches any of 20 residues, $x(i,j)$ matches i to j ($0 \leq i \leq j$) residues, $*$ matches any combination of residues with undefined length. The *support* of a pattern in a database is the ratio of sequences that satisfy the pattern over all sequences. Note that if a pattern occurs twice or more in a sequence, it is only counted once.

PGOneI

The inputs of the algorithm are dataset S which consists of a series of non-empty sequences, a minimum support threshold *min_support*, the minimal number of non-wildcard items in the patterns to be reported *min_non_wc*, and the maximal number of consecutive wildcards *max_wc_l*.

The output of the algorithm consists of all patterns that have support larger than or equal to *min_support*, and that satisfy the other input parameters. The supports are also given. These patterns are called *frequent*.

The algorithm is as follows. Here a is a pattern and S_a is the so-called projected database that contains all sequences that satisfy the pattern a , where the last element of each occurrence of a is marked (the so-called a -locations). Note that $|S_a|$, the number of sequences in S_a , is the support of a . Let *non_wc*(a) be the number of non-wildcard items in a , *curr_wc_l*(a) the number of consecutive wildcards at its end, and *last*(a) its last item.

```
PGOneI( $a, S_a$ )
  if  $|S_a| \geq \text{min\_support}$  then
    if  $\text{non\_wc}(a) \geq \text{min\_non\_wc}$  and  $\text{last}(a) \neq x$  then
      report  $a, |S_a|$ 
    for each residue  $b$  do
       $a' := a$  with  $b$  appended to it
      PGOneI( $a', S_a$ )
    if  $a \neq \Lambda$  and  $\text{curr\_wc\_l}(a) < \text{max\_wc\_l}$  then
       $a' := a$  with  $x$  appended to it
      PGOneI( $a', S_a$ )
    else PGOneII( $b, S_b$ )
```

The computation of $S_{a'}$ from S_a requires, for each a -location, the check whether or not the residue on its right side (if any) equals the newly appended item (b or x). In case of equality this gives an a' -location in $S_{a'}$. Sequences without a' -locations are deleted.

The main call is *PGOneI*(Λ, S_Λ), where Λ is the empty pattern; note that each residue in database S_Λ is a Λ -location, including the position *before* each sequence. This call creates a projected database that marks all occurrences of a single residue b , reports all frequent patterns that begin with this b , and then proceeds to the next amino acid.

PGOneII

The computation of $S_{a'}$ from S_a requires, for each a -location, the check whether or not the extension with $x(m,n)b$ matches the sequence starting from this a -location. This means that b should occur at at least $m+1$ and at most $n+1$ residues distance from the a -location. The parameter *flex* equals $n - m$.

```
PGOneII( $a, S_a$ )
  if  $|S_a| \geq \text{min\_support}$  then
    if  $\text{non\_wc}(a) \geq \text{min\_non\_wc}$  then
      report  $a, |S_a|$ 
    for each residue  $b$  do
      if  $a \neq \Lambda$  then
        for  $par = 0$  to  $flex$  do
          for  $n = par$  to  $\text{non\_wc\_l}$  do
            for  $m = n - par$  to  $n$  do
               $a' := a$  with  $x(m, n)b$  appended to it
              PGOneII( $a', S_a$ )
            else PGOneII( $b, S_b$ )
```

Note that *PGOneI* is a special case of *PGOneII* when *flex* equals 0.

PGOneIII

PGOneIII (a, S_a)

```
if  $|S_a| \geq \text{min\_support}$  then  
  if  $\text{non\_wc}(a) \geq \text{min\_non\_wc}$  then  
    report  $a, |S_a|$   
  for each residue  $b$  do  
    if  $a \neq \Lambda$  then  
      Take the first occurrence of  $b$  within the range  
      from the  $a$ -location to the corresponding  
      rightmost residue  
       $a' := a * b$   
      PGOneIII ( $a', S_a$ )  
    else PGOneII ( $b, S_b$ )
```

In this case, besides the a -locations, S_a also contains pointers to the corresponding rightmost residue, determined by the window size.