Data Mining 2: Frequent Pattern Mining Summer 2018
Christian Borgelt from 24.05.2018

Exercise Sheet 4

Exercise 14 Eclat Algorithm

a) Execute the Eclat algorithm (using the prefix tree scheme) for the transaction database shown on the right and a minimum support of $s_{\min} = 3$!

$a\ d\ f$	$a\ b\ d$
$a\ c\ d\ e$	$b\ d\ e$
$b\ d$	$b\ c\ e\ g$
$b\ c\ d$	$c\ d\ f$
$b\ c$	$a\ b\ d$

b) How is support counted in the Eclat algorithm?
How does this scheme differ from the scheme used in Apriori?
What are the advantages/disadvantages of this counting scheme?

c) Why does Eclat (usually) not exploit the Apriori property fully?
How could it be made to exploit the Apriori property fully?

d) Why is it advantageous in the Eclat algorithm to process less frequent items first?
(Hint: Consider the average length of the transaction identifier lists.)

Exercise 15 Eclat Algorithm

a) Why is a bit vector representation of the transaction identifier lists not necessarily better than an explicit list of transaction identifiers? (Hint: What happens to the transaction identifier lists in the recursive processing?)

b) In the conditional transaction databases occurring in the search for frequent item sets, it is often possible to combine transactions that have become equal after some items have been eliminated. How can this be exploited in the Eclat algorithm?

c) Show that the diffset for an item (w.r.t. some conditional database), that is, $D(a \mid I) = K(I) - K(I \cup \{a\})$, can be both larger and smaller than its cover (depending on the database).

Exercise 16 Item Order

a) How can one exploit a local item order in the Apriori algorithm?

b) How can one exploit a local item order in the Eclat algorithm?
Why is it particularly easy to use a local item order in it?

c) How can one exploit the order in which a divide-and-conquer scheme visits the item sets to optimize the reporting of found frequent item sets?
Does this also work for a local item order? Why?

Exercise 17 LCM Algorithm

a) How does the occurrence deliver scheme of LCM work?
How is it related to the Eclat algorithm? How is it different?
What additional data/information is needed for the LCM algorithm (compared to Eclat)?

b) Why does LCM allow mining with memory proportional to the database size and Eclat does not, even though their processing schemes are similar?
(Hint: Consider the order in which the subproblems are processed.)

c) Execute the LCM algorithm for the transaction database shown on the right and a minimum support of $s_{\min} = 3$!

$a\ d\ f$	$a\ b\ d$
$a\ c\ d\ e$	$b\ d\ e$
$b\ d$	$b\ c\ e\ g$
$b\ c\ d$	$c\ d\ f$
$b\ c$	$a\ b\ d$