computeMinimalTransversals\_task(toTraverselLst)

*Update maxClique, toExplore, graph\_mt from toTraverse list*

updateListFromToTraverse(toTraverse, maxClique, toExplore, graphmt

if toExplore list is not empty

*Combine maxClique list at the end of toExplore list into candidate list*

Loop on all left itemset from toExplore

Loop on all right itemset from maxClique list

call CombineItemset(leftItemset, rightItemset)

if combined itemset contains is not a clone and isEssential

add combined itemset into new ToTraverse list

if new traverse list is not empty

Launch a new thread with computeMinimalTransversals\_task(newToTraverselLst) as // task

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Itemset1 | Itemset2 | Itemset3 | Itemset4 | Itemset5 | Itemset6 | Itemset7 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Itemset1 | Itemset2 | Itemset3 | Itemset4 | Itemset5 | Itemset6 | Itemset7 |

*Combine items from itemsets and remove duplicates: "1" + "2" => "12", "71" + "72" => "712"*

CombineItemset(leftItemset, rightItemset)

Create a new combinedItemset as a copy of leftItemset

loop on all item from leftItemset

copy left itemset into combined itemset

Loop on all items from rightItemset

*Remove duplicates*

If right item is not present into combined itemset

*Update OR value for disjunctive support*

Update OR value of combined itemset

*Add the right item*

Add right item into combined itemset list

Update disjunctive support value

Update OR support of combined itemset

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Update maxClique, toExplore, graph\_mt from toTraverse list*

*Compute disjunctive support for each itemset of toTraverse list*

*if disjunctive support is equal to object count, add the itemset to graphMt list (then process its clones)*

*if not, combine itemset one by one from toTraverse list*

*if disjunctive support for combined itemset is equal to object count, add the itemset to toExplore list*

*if not, add the itemset to maxClique list*

updateToTraverseList(toTraverse, maxClique, toExplore, graphMt)

loop on all “current itemset” from toTraverselist itemset

compute disjonctive support

if disjonctive support == object count

add current itemset to graphMt

loop on all “current item” from current itemset

call recurseOnClonedItemset(current itemset, current itemset, graphMt)

if “current item” is an original from a clone

loop on all “clone item” from current item

*create a cloned itemset as a copy of current itemset and replace current item by clone item*

loop on all “current item” from current itemset

if “current item” != clone item”

copy current itemset into cloned itemset

else

copy clone item into cloned itemset

add cloned itemset to graphMt

loop on all “current item” from cloned itemset

recurseOnClonedItemset(cloned item set, current item, graphMt)

else if “current itemset” is the 1st of toTraverseList and size of “current itemset” == 1

add current itemset to maxClique list

store current itemset as previous itemset

else

call combineItemset(previous itemset, current itemset)

*compute disjonctive support from combined itemset*

if discjonctive support != object count

add current itemset to maxClique list

or

add current itemset to toExplore list

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

isEssential(itemset)

if isEssential has one item

return true

isEssential = false

loop all “item1” from itemset

*compute or sum of all item from the itemset*

loop all “item2” from itemset

if “item1” != “item2”

sum = sum | item2

loop on object count 🡪 i

get ith bit of sum [0, object count] 🡪 bit0

get ith bit of item1 🡪 bit1

if(!bit0 && bit1)

isEssential = true

break loop on object count

if !isEssential

break loop on itemset

return isEssential