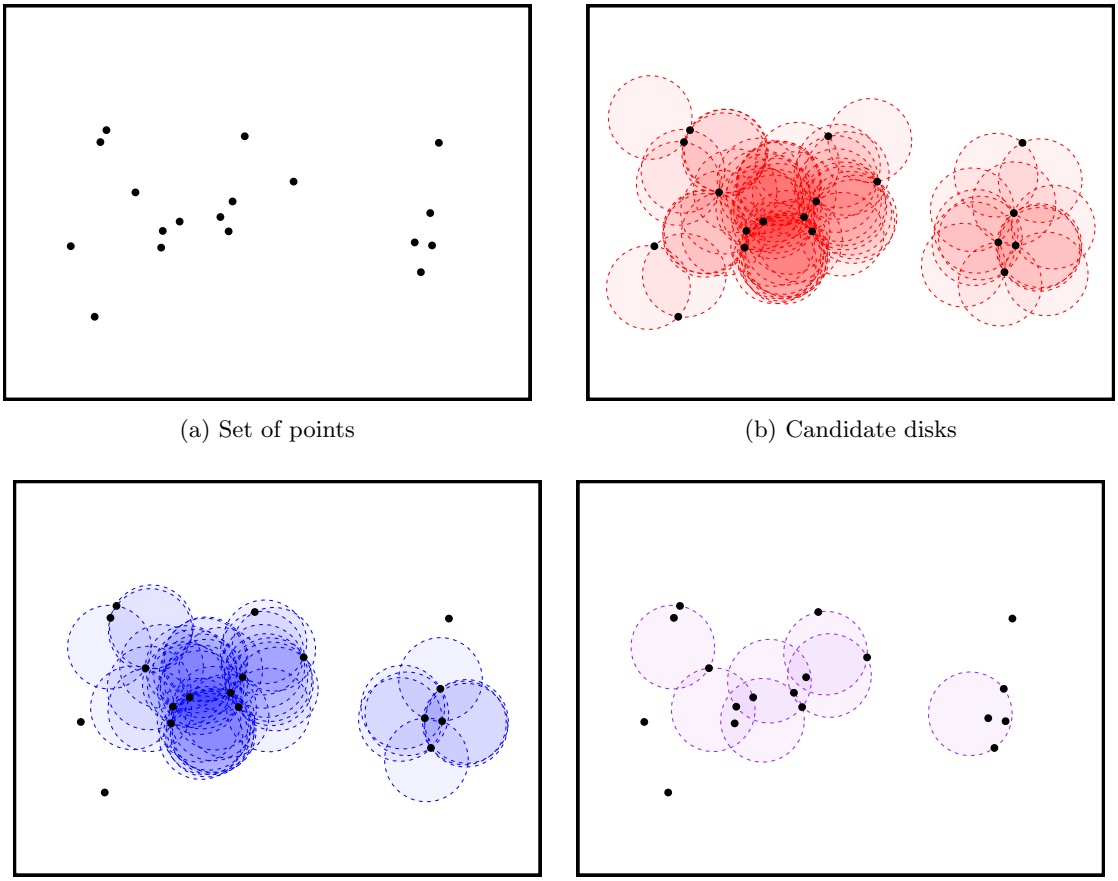

(a) Set of points

(b) Candidate disks

(c) Removing less than μ disks

(d) Removing redundant disks

Figure 1: Steps during disk filtering ($\mu = 3$).

Improving disk filtering by maximal pattern detection.

After the candidate disks detection, additional filtering should be carried. BFE algorithm states two filtering process: Firstly, remove any disk which do not group a minimum number of moving object (defined by μ parameter). Secondly, remove any disk whose element set is a subset of another disk's element set. It means that If from two disks one of them contains all the other's elements, just the former should be kept. If two disks share the same elements, just one of them should remain. Figure 1 illustrate the steps.

At this point, it is assumed that a valid set of candidates disk has been generated. It is a simple list with a generated key for each disk and a list with the ID's of the moving objects which the disk encloses. The coordinates of the disk are not needed for filtering purposes but they should be useful if further visualizations are planned.

The task to obtain a valid set of disks from the candidate set is not trivial. A brute force algorithm takes $O(n^2)$ complexity to prune redundant disks which is not scalable for big spatial datasets. One approach is see the elements inside each disk as a set or transaction. Once a transactional version of the candidate disks is obtained, it can be analyzed with well-know algorithms for supersets detection.

Maximal and closed frequent patterns

The detection of supersets in frequent pattern mining has been a widely discussed topic. Previous works has shown the advantage of maximal and closed frequent patterns as valid representations of the full set of frequent patterns in transactional databases. The main idea is to find the most general pattern from which the remain set of pattern can be generated.

Although the first generation of algorithms was designed to find the complete group of frequent itemsets, in large databases using low values for minimum support threshold this number can be huge. This is because if an itemset is frequent, each of its subsets is frequent as well. Long itemsets will contain large number of shorter frequent subsets. For instance, let a long itemset $I = \{a_1, a_2, ..., a_{100}\}$ with 100 items (It is usually called type 100 or 100-itemset for its number of members).

It will contain $\binom{100}{1}$ 1-itemsets, $\binom{100}{1}$ 2-itemsets, and so on. The total number of frequent itemsets that it would contain would be:

$$\binom{100}{1} + \binom{100}{1} + ... + \binom{100}{100} = 2^{100} - 1 \approx 1.27 \times 10^{30}$$

To overcome this drawback the concepts of *closed frequent pattern* and *maximum frequent pattern* are used. A pattern α is a *closed frequent pattern* if α is frequent and there exists no other pattern, with the same support, whose contains α. On the other hand, a pattern α is a *maximal frequent pattern* if α is frequent and there exists no other pattern, with any support, whose contains α. For example:

$\alpha = \{a_1, a_2, a_3, a_4 : 2\}$ α is *maximal*
$\beta = \{a_1, a_2, a_3 : 4\}$ β is *closed* but not *maximal*

The set of maximal frequent patterns is important because it contains the set of longest patterns such that any kind of frequent pattern which exceeds the minimum support can be generated.For clarification, these two concepts can be illustrated with an additional example. Suppose a database D contains 4 transactions:

$$D = \{\langle a_1, a_2, ..., a_{100}\rangle; \langle a_1, a_2, ..., a_{100}\rangle; \langle a_{20}, a_{21}, ..., a_{80}\rangle; \langle a_{40}, a_{41}, ..., a_{60}\rangle\}$$

Note that the first transaction is repeated twice. The minimum support $min\_sup = 2$. A complete search for all itemsets will generate a vast number of combinations. However, the closed frequent itemset approach will find only 3 frequent itemsets:

$$C = \{\{a_1, a_2, ..., a_{100} : 2\}; \{a_{20}, a_{21}, ..., a_{80} : 3\}; \{a_{40}, a_{41}, ..., a_{60} : 4\}\}$$

The set of closed frequent itemsets contains complete information to generate the rest frequent itemsets with their corresponding support. It is possible to derive, for example, $\{a_{50}, a_{51} : 4\}$ from $\{a_{40}, a_{41}, ..., a_{60} : 4\}$ or $\{a_{90}, a_{91}, a_{92} : 2\}$ from $\{a_1, a_2, ..., a_{100} : 2\}$. On the other hand, we just obtain one maximal frequent pattern, in this case:

$$M = \{\{a_1, a_2, ..., a_{100} : 2\}\}$$

This only pattern will serve as a valid representation of all the frequent patterns as it contains all the possible subsets.

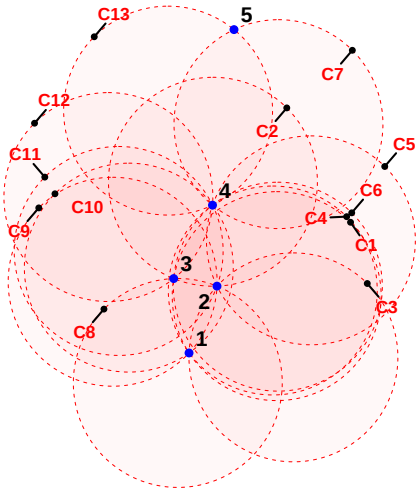

Figure 2: Towards a transactional version of the candidate disks

A transactional version of the candidate disks

The main idea to deal with the candidates disks as a set of transactions is the possibility to apply well-known algorithms for maximal pattern detection in order to filter all the possible subsets. In figure 2 we can see that we can extract the points enclosed by each disks in the form of transaction. For example:

$$
\begin{array}{lll}
D = & \{\langle 1,2,3\rangle; & \textit{From disk 1} \\
& \langle 2,3,4\rangle; & \textit{From disk 2} \\
& \langle 1,2\rangle; & \textit{From disk 3} \\
& \vdots & \vdots \\
& \langle 1,2,3,4\rangle; & \textit{From disk 10} \\
& \langle 2,3,4\rangle; & \textit{From disk 11} \\
& \langle 3,4\rangle; & \textit{From disk 12} \\
& \langle 4,5\rangle\} & \textit{From disk 13}
\end{array}
$$

In this way, it is possible to apply any algorithm for maximal pattern detection to discard redundant disks by using $min\_sup = 1$. The result should be:

$$
\begin{array}{ll}
M = & \{\langle 1,2,3,4\rangle; \\
& \langle 4,5\rangle\}
\end{array}
$$

From this shorter list we should prune $\langle 4,5\rangle$ because it does not fulfill the $\mu = 3$ parameter. As an advantage on the use of maximal pattern algorithms we could expect a significant improvement in performance since many of this techniques are able to find the set of maximal disk in linear time according with the number of maximal patterns.

Proof by contradiction

Proof. Given two disks in the initial set of candidate disks we have to consider two option: Firstly, let's have a disk A whose elements are $A = \{p_1, p_2, p_3, ..., p_n\}$ and a disk B which have the same elements $B = \{p_1, p_2, p_3, ..., p_n\}$ so $A = B$. If they are the input of a maximal pattern algorithm it will choose any of them, let say B and A will be consider a redundant disk and it can be safely discarded. Secondly, let's a disk A whose elements are $A = \{p_1, p_2, p_3, ..., p_n\}$ and a disk B whose elements are $B = \{p_1, p_2, p_3, ..., p_n, p_{n+1}, ..., p_m\}$ so $B \supset A$. If they are the input of a maximal pattern algorithm it will choose disk B and A will be consider a redundant disk and it can be safely discarded. □