Duplicate detection of 2D-NMR Spectra

Alexander Hinneburg

Björn Egert and Andrea Porzel

Institute of Informatics, Martin-Luther-University Halle-Wittenberg, Halle/Saale, Germany,
hinneburg@informatik.uni-halle.de

Leibniz Institute of Plant Biochemistry, Halle/Saale, Germany, {begert,aporzel}@ipb-halle.de

Abstract

2D-Nuclear magnetic resonance (NMR) spectra are used in the (structural) analysis of small molecules. In contrast to 1D-NMR spectra, 2D-NMR spectra correlate the chemical shifts of ^{1}H and ^{13}C at the same time. A spectrum consists of several peaks in a two-dimensional space. The most important information of a peak is the location of its center, which captures the bonding relationships of hydrogen and carbon atoms. A spectrum contains much information about the chemical structure of a product, but in most cases the structure cannot be read off in a simple and straightforward manner. Structure elucidation involves a considerable amount (manual) efforts.

Using high-field NMR spectrometers, many 2D-NMR spectra can be recorded in short time. So the common situation is that a lab or company has a repository of 2D-NMR spectra, partially annotated with the structural information. For the remaining spectra the structure in unknown. In case two research labs are collaborating, the repositories will be merged and annotations shared.

We reduce that problem to the task of finding duplicates in a given set of 2D-NMR spectra. Therefore, we propose a simple but robust definition of 2D-NMR duplicates, which allows for small measurement errors. We give a quadratic algorithm for the problem, which can be implemented in SQL. Further, we analyze a more abstract class of heuristics, which are based on selecting particular peaks. Such a heuristic works as a filter step on the pairs of possible duplicates and allows false positives. We compare all methods with respect to their run time. Finally we discuss the effectiveness of the duplicate definition on real data.

1 Motivation

Nuclear magnetic resonance (NMR) spectra are important for analyzing unknown natural products. In contrast to standard one-dimensional NMR spectroscopy, advanced two-dimensional NMR spectroscopy is able to capture the influences of two different atom types at the same time (e.g. ^{1}H, hydrogen and ^{13}C carbon).

The result of an 2D-NMR experiment is an intensity function measured over two independent variables[1]. Regions of the plane with high intensity are called peaks, which contain the real information about the underlying molecular structure. Modern devices are able to produce very strong magnetic fields and produce spectra with sharp peaks on the two-dimensional plane. The

[1]The measurements are in parts per million (ppm).

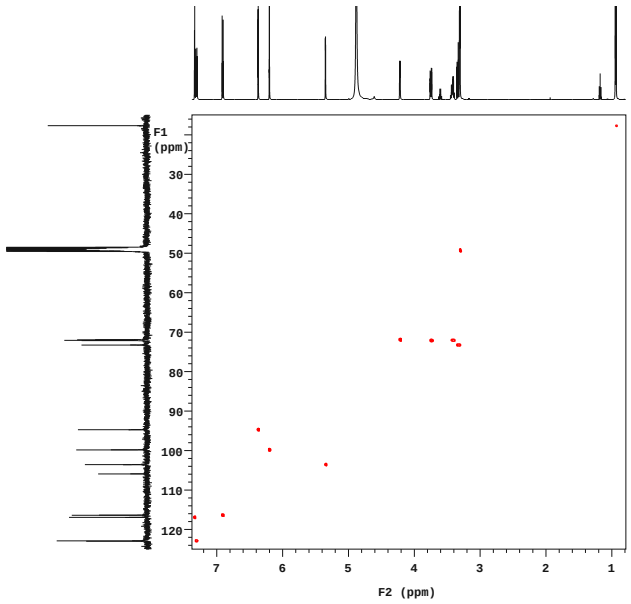

Figure 1: 2D-NMR spectrum of Quercetrin, the one-dimensional plots at the axes are projections of the original two-dimensional intensity function.

usual visualizations of 2D NMR spectra are contour plots as shown in figure 1. A threshhold removes contour lines in low intensity regions, because they are produced by random fluctuations. An ideal peak would be represented as small dot. However, if only a limited resolution is available (depending on the strength of the magnetic field) multiple peaks may appear as a single merged object with non-convex shape. In contrast to 1D- Spectra the second dimension diminishs the problem of high intensity peaks masking nearby low intensity signals.

The peak pattern is very characteristic and specific for a particular substance. In the biochemical literature, peaks are noted by their two-dimensional position without any information about the shape of the peak. Thus, we represent a peak of a spectrum as a two-dimensional point. However, in a spectrum measured with low resolution two different peaks, which are close together, may be merged and so both are represented by a single point.

Since modern NMR devices allow the automatic analysis of many probes per day, the number of spectra in a database can be up to several thousands. Because deducing the chemical structure of a complex organic substance from the spectrum is still difficult, most of the NMR data is unpublished – but contains a lot of experimental knowledge. We face the following situation that on the one hand we have a lot of experimental data measured automatically and on the other hand for only a small fraction of the spectra the chemical structure is known. Data mining can help to relate spectra for which the chemical structure is unknown to those which have already been investigated.

One problem is to find whether a substance has already been examined and the structure is known. As the NMR spectra can be measured with different resolution, some of the peaks may be merged in the spectrum with lower resolution and the task of finding duplicates is not trivial. A method which is able to find duplicates could be used for data integration in case two or more labs share their data.

Our contributions in this paper are:

- We propose a simple but robust definition of 2D-NMR duplicates, which allows for small

measurement errors.

- We give a quadratic algorithm which can be implemented in SQL.
- We develop a more abstract class of heuristics, which act as a filter for possible pairs of duplicates.
- All methods are implemented in standard SQL, which allow an easy dissemination of the technology.

The remainder of the paper is organized as follows: First we discuss some related work in section 2. Then we introduce a simple definition of similarity, which in turn is used to define fuzzy duplicates. In section 4 we discuss properties of our duplicate definition and introduce our methodology for finding duplicates in section 5. In section 6, we conduct experiments with real data and with section 7 we conclude the paper.

2 Related Work

Duplicate detection can be seen as a special case of content-based similarity search. Pairs of duplicates are spectra, whose similarity exceeds a certain cutoff value. While content-based similarity search is already in use for 1D-NMR spectra [11,1,15,10,2], to the best of our knowledge, no effective similarity search method is known for 2D-NMR-spectra. Besides technical details, such as the choice of a particular cutoff value, the problem of an approach purely based on similarity is, that the similarity between all pairs of spectra has to be computed. This leads to a run time that is quadratic in the number of spectra, which is prohibitive for large spectra databases. In case of duplicate detection, more efficient algorithms are needed.

Various aspects of detecting duplicates have received attention in the past database- and information retrieval research, such as near-duplicate detection of documents. The efficient detection of near-duplicate documents has been studied by several authors [4, 17]. In particular near-duplicate detection of web documents is an quite active research area [8, 5, 7]. The difference between near-duplicate documents and fuzzy duplicates of 2D-NMR spectra is that documents are composed of discrete entities, namely words or index terms, whereas 2D-NMR spectra consist of continuous 2D points. The crucial difference is that the matching operation for words is transitive but not for 2D points. An extension of near-duplicate documents are duplicates in XML documents [16], where the set of terms is organized as tree.

Duplicates are often picked using a similarity measure. Such measures can be defined manually, but for strings suitable similarity measures can be learned automatically using support vector machines [3], which improves the detection accuracy. Searching the WHO drug safety database for duplicates is one example of difficult duplicate detection. [13].

Since duplicates are hard to detect in first place, it seems difficult to find subquadratic algorithms for this problem class. Fortunately, fuzzy duplicates of 2D-NMR spectra have a simple definition, which does not require advances learning techniques.

The detection of duplicate records in data streams [6] or click streams [12] are new variants of the problem. Here, the duplicates have simple definitions and the records have fixed length. NMR spectra do not have such a simple nature, e.g. the number of peaks differ between spectra.

The techniques, especially Bloom filters, might be applicable to NMR spectra as well, and future research will shown how they perform in in our scenario.

Finally, the detection of duplicates in images [9] is slightly related to our research, since 2D-NMR spectra could be considered to be images as well. However, the used techniques e.g. in [9] ensure invariance wrt. scaling, shifting and rotation, which is not meaningful in case of 2D-NMR spectra.

3 Definition of Similarity and Fuzzy Duplicates

A two-dimensional spectrum of an organic substance captures many chemical characteristics such as rings and chains.

Definition 1 *A 2D-NMR spectrum A is defined as a set of points $\{x_1, \ldots, x_n\} \subset \mathbb{R}^2$. The $|\cdot|$ function denotes the size of the spectrum $|A| = n$.*

The number of peaks per spectrum in our database is typically between 4 and 30. Our definition of duplicates is based on the idea that peaks can be matched. Since spectra are measured experimentally, the peak positions differ even between repeated measurements. Therefore, peaks cannot be matched using their exact positions, but some deviation must be allowed. A simple and effective approach is to require that a peak matches another peak within a certain spatial neighborhood. The neighborhood is defined by the ranges α and β.

Definition 2 *A peak x from spectrum A **matches** a peak y from spectrum B, iff $|x.c - y.c| < \alpha$ and $|x.h - y.h| < \beta$, where $.c$ and $.h$ denote the NMR measurements for carbon and hydrogen respectively.*

Based on the notion of matching peaks, we can define a set-oriented similarity measure, from which in turn we derive the definition of duplicates as a special case. Note, that a single peak of a spectrum can match several peaks from another spectrum. Given two spectra A and B, the subset of peaks from A for which exist matching partners in B, is denoted as $matches(A, B) = \{x \colon x \in A, \exists y \in B \colon x \text{ matches } y\}$. The function $matches$ is not symmetric, but can be used to define a symmetric similarity measure

Definition 3 *Let be A and B two spectra and $A' = matches(A, B)$ and $B' = matches(B, A)$, so the **similarity** is defined as*

$$sim(A, B) = \frac{|A'| + |B'|}{|A| + |B|}$$

$sim(A, B)$ is close to one if most peaks of both spectra are matching peaks. Otherwise, the similarity drops towards zero.

An important application of similarity search is the detection of duplicates to increase the data quality of a collection of 2D-NMR-spectra. Clearly, a naïve definition of duplicates that requires two duplicate spectra A and B to have the same size and the peaks at the same positions misses many duplicates.

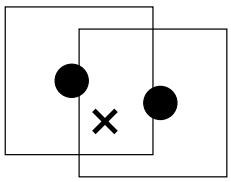

Figure 2: The peak x (cross) from spectrum A is matched by the two peaks y_1 and y_2 (dots) from spectrum B.

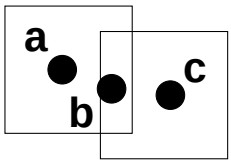

Figure 3: The peak a from spectrum A matches peak b from spectrum B and b matches c from spectrum C. However a and c are not matching.

The reason is that the spectra are measured experimentally and peak positions differ even if the same probe is analyzed twice. So flexibility should be allowed for the peak positions. Another problem occurs if two spectra of the same substance are measured with different resolutions. In case a spectrum is measured with low resolution it may happen that neighboring peaks are merged to a single one. These cases can not be handled if the matches are restricted to an one-to-one relationship, and that a single peak from spectrum A can be matching partner for two close peaks from spectrum B (see figure 2).

We propose a definition of fuzzy duplicates based on the similarity measure, which can deal with both problems mentioned, namely deviances in peak measurements as well as merged peaks.

Definition 4 *A pair of 2D-NMR-spectra A and B are **fuzzy duplicates**, iff $sim(A, B) = 1$.*

With this definition it is only required that every peak of a spectrum finds at least one matching peak in the other spectrum. The parameters α and β can be chosen with application knowledge of typical variances of single peak measurements. In our application, we chose $\alpha = 3$ ppm (^{13}C coordinate) and $\beta = 0.3$ ppm (^{1}H coordinate) unless stated otherwise. To account for very different experimental conditions (solvent, pH- value) the tolerances have to be adapted.

4 Theoretical Complexity Considerations

Our definition has several properties: First, the duplicate definition is symmetric, that means if spectrum A is a duplicate for spectrum B than B is also a duplicate for A. At a first glance there is nothing spectacular about this, but this property will prove useful in section 5.2. The second observation is that our duplicate definition is not transitive, that means if A is duplicate for B and B is duplicate for C that not necessarily A is duplicate for C. An example for this fact is sketched in figure 3. The reason is the nature of continuous measurements of the peak coordinates.

The lack of transitivity has the consequence that a set of duplicate spectra (all pairs of spectra of the set are duplicates) cannot be represented by a single spectrum, which would ease the

detection of duplicates, since all spectra which are duplicates of the representative are also pairwise duplicates. Because fuzzy duplicates of 2D-NMR spectra do not have this property, all pairs of the set have to be checked in order to comprise a set of duplicates. Thus, the complexity of an algorithm that detects all duplicates in a set of spectra has a quadratic worst case run time. Therefore we have to resort to heuristics which reduce the average runtime on typical data sets. A general method to derive such heuristic is to find necessary conditions which can quickly eliminate many pairs of possible duplicates. The remaining pairs have to be checked with the original definition of duplicates in order to eliminate false duplicates.

5 Finding Duplicates

5.1 The Naïve Method

Given the definition of duplicates we can search the data to eliminate them. The straight forward implementation of the duplicate definition is to check every pair of spectra, whether the pair is covered by the duplicate definition or not. Such a pair checking can be formulated as SQL query. We assume the spectra to be stored in a relational database with the following schema:

```
spectrum(spectrumId, name, n)
peak(spectrumId,peakId,c,h)
```

In order to allow fast access to the tuples in the tables the attributes spectrumID and peakID are indexed. The peaks of a spectrum A are represented by $n = |A|$ tuples in the peak table. An example for a query to find all duplicate with $\alpha = 3$ ppm and $\beta = 0.3$ ppm is shown below.

```
SELECT A.spectrumId, A.name, A.n,
       B.spectrumId, B.name, B.n
FROM spectrum A, spectrum B
where
 A.spectrumId>B.spectrumId AND
 A.n=
   (
   select count(*)
   from peaks PA
   where PA.spectrumId = A.spectrumId AND
    1<= ( select count(*)
          from peaks PB
          where PB.spectrumId=B.spectrumId AND
           ABS(A.c - B.c) < 3 AND
           ABS(A.h - B.h) < 0.3
         )
   ) AND
 B.n=
   (
   select count(*)
   from peaks PB
```

```
where PB.spectrumId = B.spectrumId AND
 1<= ( select count(*)
       from peaks PA
       where PA.spectrumId=A.spectrumId AND
        ABS(A.c - B.c) < 3 AND
        ABS(A.h - B.h) < 0.3
     )
)
```

5.2 Framework for fast Duplicate Detection

To reduce the complexity of the naïve method, we need to develop heuristics which (i) quickly check necessary conditions of a pair of spectra to be a duplicate and (ii) avoid false negatives so that no duplicate pairs are missed. A single heuristic may not reduce the number of candidate pairs significantly. Thus, several heuristics can be combined by requiring that all heuristic checks are true, which is a logical AND.

In general, the output of a heuristic is a set of candidate pairs which passed the test of some necessary condition for duplicates. Given the set of spectra S containing m spectra, such an output can be represented by a bit matrix $F \in \{0,1\}^{m \times m}$. In order to reduce the number of candidate pairs several heuristics can be combined. The combination of several heuristics with outputs $F^1, \ldots F^k$ is a bit matrix F with the elements $F_{ij} = F^1_{ij} \wedge \ldots \wedge F^k_{ij}$. Such a bit matrix usually contains many zeros and a few ones, so it can be stored as a sparse matrix. Intermediate results during the combination of outputs of several heuristics should be kept small. Therefore the processing order should go from the matrix with the smallest number of ones to the matrix with the largest number of ones.

How to find heuristics which implement necessary conditions for a pair being a duplicate in a systematic way? A simple strategy to generate such heuristics is to select a peak x from a spectrum A and search for neighboring peaks of the selected peak. The spectra which have a peak in the neighborhood of x are candidates for being a duplicate of A. All other spectra cannot be duplicates for A and are therefore excluded from further examination. We call such heuristics peak selecting heuristics.

Our definition of duplicates is symmetric, that means when A is duplicate of B then B is also a duplicate of A. In general, the output of a peak selecting heuristic is not symmetric, because the peak selected from A is not necessarily in the neighborhood of the peak selected from B. There are two options to deal with symmetry. First, symmetry can be used as additional filter. In case, $F_{ij} = 1$ and $F_{ji} = 0$ the one of F_{ij} can be zeroed. The second option is to ignore symmetry and compute only one half of the bit matrix, such as the upper triangle. The questions is whether the costs to check the additional candidates being generated by ignoring symmetry is lower than the costs to compute the lower triangle bit matrix. We argue that it is beneficial to check a few more candidates to save the costs for the lower triangle bit matrix.

If half of the bit matrix is not computed, the question arises whether we can decrease the number of ones in the upper triangle matrix by selecting a special permutation of rows and columns. The costs to find such a permutation should be small, as it is only beneficial, if those costs pay off by the costs for the saved candidates. Two permutations are interesting candidates: the first permutation is to sort the spectra by the number of peaks in increasing order. Thus, rows

corresponding to spectra with many peaks are at the bottom of the matrix and do not contribute much to the upper triangle of the matrix. The second permutation is to sort the spectra by the sum of the densities of their peaks in increasing order: rows corresponding to spectra with peaks in high density regions in the peak space are at the bottom of the matrix and do not contribute much to the upper triangle of the matrix. The density in the peak space can be estimated by a two-dimensional histogram.

In case of peak selecting heuristics, the optimal peak we can select for each spectrum is a peak with the minimal number of matching peaks from other spectra. The set of spectra having a matching peak for the peak $x \in A$ is denoted by

$$N(x) = \{B \in S\colon A \neq B, \exists x' \in B \text{ with } |x'.c - x.c| \leq \alpha \text{ and} |x'.h - x.h| \leq \beta\}$$

An optimal peak $x \in A$ has the property $|N(x)| = \min_{x' \in A}\{|N(x')|\}$. Thus, in order to be able to select an optimal peak, it is required to know the total set of spectra S. In contrast to the optimal peak selecting heuristic, the simplest peak selecting heuristic is to select a peak at random from each spectrum. This heuristic will serve as base line condition. Other possible heuristics include

Minimal C-Shift Select the peak $x \in A$ with $x.c = \min_{x' \in A}\{x'.c\}$

Minimal H-Shift Select the peak $x \in A$ with $x.h = \min_{x' \in A}\{x'.h\}$

Maximal C-Shift Select the peak $x \in A$ with $x.c = \max_{x' \in A}\{x'.c\}$

Maximal H-Shift Select the peak $x \in A$ with $x.h = \max_{x' \in A}\{x'.h\}$

5.3 SQL Implementation of the Heuristics

In this section we describe our implementation of the heuristics in standard SQL on ORACLE 10g, but the statements can execute on other standard database management systems like IBM DB2 or Postgres as well.

The general idea of peak selecting heuristics is to select a peak from a spectrum with a particular property and find all spectra which have a matching peak for the selected one. Those spectra are the duplicate candidates which have to be checked by the duplicate definition.

The optimal peak selecting heuristic minimizes the length of the candidate list for a spectrum by determining for each spectrum the peak with the smallest number of neighboring peaks. The geometric approach would be to issue for each peak of a spectrum a region query to retrieve the peaks in the neighborhood. Such queries can be efficiently supported by R*-trees or a similar spatial index structures. In our application setting, the peaks are two-dimensional points and the neighborhood definition is the same for all peaks. So, the use of a complex index structure like the R*-tree seems overkill, since the R*-tree also has to be build first, which may involve large costs. A second disadvantage is that such index structures are available for most database systems only as add-ons. Therefore we focus on a simpler solution based on standard SQL 99.

Our idea is inspired by average shifted histograms [14], where the point density in a space is estimated by shifted grids. We formalize a two-dimensional grid as a function which takes two

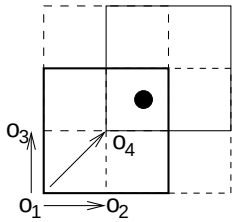

Figure 4: A peak with the four cells from the different grids. The cell with $o_1 = (0, 0)$ has bold lines, $o_2 = (\alpha, 0),\ o_3 = (0, \beta)$ have dashed lines and $o_4 = (\alpha, \beta)$ has normal lines.

real numbers and delivers a pair of integers. An equidistant grid depends on the sizes of the intervals along the dimensions $w_c, w_h \in \mathbb{R}$ and the origin $o \in \mathbb{R}^2$ of the grid. In general, the grid function can be denoted like:

$$x, o \in \mathbb{R}^2,\ g(x) = (g_c, g_h),\ \text{with } g_c = \left\lfloor \frac{x.c - o.c}{w_c} \right\rfloor,\ g_h = \left\lfloor \frac{x.h - o.h}{w_h} \right\rfloor$$

All points in a grid cell are mapped to the same integer coordinates. The ideal case occurs if a peak is placed exactly at the center of a grid cell, then the cell with the widths $w_c = 2 \cdot \alpha$ and $w_h = 2 \cdot \beta$ respectively includes all matching peaks defined by α and β. However, since a peak may be placed anywhere within the cell, it captures only a fraction of the peaks' neighborhood. Our idea is to use four shifted grids, where a single peak is member of four grid cells. The shifts of the grids are determined such that the union of all four grid cells includes the whole neighborhood. The situation is illustrated in figure 4, which shows the four cells which include a given peak.

Thus, we obtain the following property. For any $o \in \mathbb{R}^2$ the shifted origins for the grids g_1, g_2, g_3 and g_4 are $o_1 = o + (0, 0),\ o_2 = o + (\alpha, 0),\ o_3 = o + (0, \beta)$ and $o_4 = o + (\alpha, \beta)$. Then for any point $q \in \mathbb{R}^2$ the neighborhood is included in the union of the four cells which contain q:

$$\begin{aligned}\big\{x \in \mathbb{R}^2 \colon |q.c - x.c| < \alpha \text{ and } |q.h - x.h| < \beta\big\} \subseteq \\ \big\{x \in \mathbb{R}^2 \colon g_1(q) = g_1(x)\ \vee\ g_2(q) = g_2(x)\ \vee\ g_3(q) = g_3(x)\ \vee\ g_4(q) = g_4(x)\big\}\end{aligned}$$

That property allows to approximate the neighborhoods of all peaks with grids. Grids can be implemented in typical databases using the ROUND function together with the GROUP BY clause. To save recalculations of the grid coordinates we added appropriate columns to the peak table. Using the choices for $\alpha = 3$ ppm and $\beta = 0.3$, the altered peak table is defined as:

```
peak(spectrumId,peakId,c,h, g1c,g1h,g2c,g2h,g3c,g3h,g4c,g4h)

update peaks set g1c=floor(c/6.0)
update peaks set g1h=floor(h/0.6)
update peaks set g2c=floor((c+3.0)/6.0)
update peaks set g2h=floor(h/0.6)
update peaks set g3c=floor(c/6.0)
update peaks set g4h=floor((h+0.3)/0.6)
update peaks set g4c=floor((c+3.0)/6.0)
update peaks set g4h=floor((h+0.3)/0.6)
```

All of the discrete integer attributes are indexed. For better readability and to reduce recalculations we define views for intermediate results. In the first view we determine for the peaks of all spectra the number of neighboring peaks.

```
create view spec_peak_no as
select a.spectrumId, a.peakId,
       sum(peaksPerCell) as peakNo
from peaks a, (
  select g1c, g1h, g2c, g2h,g3c, g3h, g4c, g4h,
         count(*) as peaksPerCell
  from peaks
  group by g1c, g1h, g2c, g2h, g3c, g3h, g4c, g4h
  ) b
where (a.g1c=b.g1c and a.g1h=b.g1h) or
      (a.g2c=b.g2c and a.g2h=b.g2h) or
      (a.g3c=b.g3c and a.g3h=b.g3h) or
      (a.g4c=b.g4c and a.g4h=b.g4h)
group by  a.spectrumId, a.peakId
```

In a second step we determine for each spectrum the peak with the minimal number of neighboring peaks. In case the minimum is not unique we take the peak with the lowest peakId.

```
create view spec_min_peak as
select a.spectrumId, min(b.peakId) as peakId,
       a.minPeakNo
from (
     select spectrumId, min(peakNo) as minPeakNo
     from spec_peak_no
     group by spectrumId
     ) a, spec_peak_no b
where a.spectrumId=b.spectrumId and
      a.minPeakNo=b.peakNo
group by a.spectrumId, minPeakNo
```

In the last step we generate a unique list of pairs of spectrumId's which have to be checked whether they are covered by the duplicate definition, based on the neighborhoods of the selected peaks:

```
create view compare_spec_multiple as
select x.spectrumId as spectrumId1, x.n as n1,
       y.spectrumId as spectrumId2, y.n as n2
from (
  select distinct a.spectrumId as s_id1,
                  b.spectrumId as s_id2
  from spec_min_peak_multiple a, peaks a_help, peaks b
  where a.spektrum_id=a_help.spektrum_id and
    a.peak_id=a_help.peak_id and
    a.spectrumId < b.spectrumId and
```

```
    ((a_help.g1c=b.g1c and a_help.g1h=b.g1h) or
     (a_help.g2c=b.g2c and a_help.g2h=b.g2h) or
     (a_help.g3c=b.g3c and a_help.g3h=b.g3h) or
     (a_help.g4c=b.g4c and a_help.g4h=b.g4h))
  ) cs, spectrum x, spectrum y
where cs.s_id1=x.spectrumId and
      cs.s_id2=y.spectrumId
```

The last view replaces the self join of the spectrum table (line 3) in the SQL statement for the naïve method. The difference is that the number of pairs which have to be checked is much smaller.

The heuristics minimal-C shift, maximal C shift, minimal H shift, and maximal H shift are simpler to implement. The view to create the candidates based on minimal-C shift is as follows.

```
create view compare_spec_minC as
select  Aspec.spectrumId, Aspec.n, Bspec.spectrumId, Bspec.n
from (
  select distinct P.spectrumID as AspectrumID,
                  P2.spectrumID as BspectrumID
  from (
       select P.spectrumID, min(P.peakID) min_peakID
       from ( select P.spectrumID as spectrumID, min(c) as minc
              from peaks P
              group by P.spectrumID
             ) P_min_c, peaks P
       where P_min_c.spectrumID=P.spectrumID AND
             P.c=P_min_c.minc
       group by P.spectrumID
     ) P_peak, peaks P, peaks P2
  where P_peak.spectrumID=P.spectrumID AND
        P.spectrumID < P2.spectrumID AND
       P_peak.min_peakID=P.peakID AND
      (abs(P.c-P2.c)<=3 AND abs(P.h-P2.h)<=0.3)
) C, spectrum Aspec, spectrum Bspec
where
 C.AspectrumId=Aspec.spectrumId AND C.BspectrumId=Bspec.spectrumId
```

The views for the other three heuristics are analogous. For the random heuristic a function is needed to generate random integers. Oracle provides DBMS_RANDOM.VALUE(1,n), which delivers a random float number between 1 and n. The view for that heuristic is

```
create view compare_spec_rand as
SELECT Aspec.spectrumId, Aspec.n, Bspec.spectrumId, Bspec.n
FROM (
select distinct P.spectrumID as AspectrumID,
                P2.spectrumID as BspectrumID
from (
       select P.spectrumID, FLOOR(DBMS_RANDOM.VALUE(1,P.n))
```

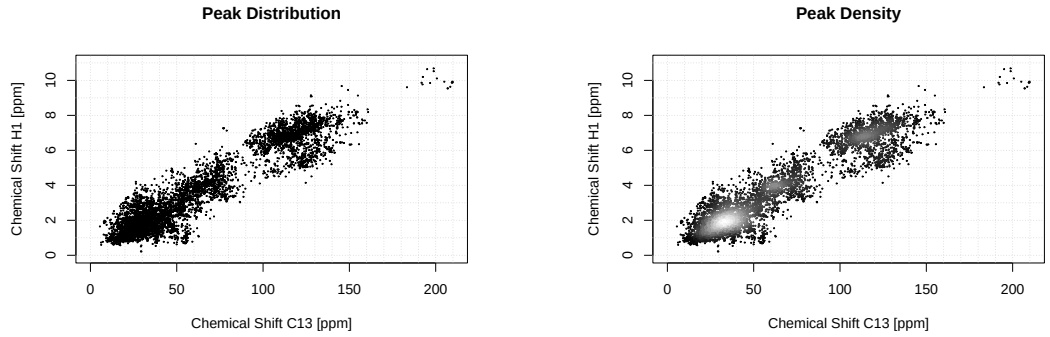

Figure 5: Distribution (left) and density (right) of the peaks of all spectra. Light gray means higher density.

```
            as rand_peakID
       from spectrum P
     ) P_peak, peaks P, peaks P2
where P_peak.spectrumID=P.spectrumID AND
      P.spectrumID < P2.spectrumID AND
      P_peak.rand_peakID=P.peakID AND
      (abs(P.c-P2.c)<=3 AND abs(P.h-P2.h)<=0.3)
) C, spectrum Aspec, spectrum Bspec
where
 C.AspectrumId=Aspec.spectrumId AND C.BspectrumId=Bspec.spectrumId
```

6 Experiments

In this section we evaluate the proposed definition of duplicates and conduct experiments to investigate the tradeoff between the costs to check a certain number of candidate pairs for duplication and the costs to determine those candidate pairs.

6.1 2D-NMR Data

The substances included in the database are mostly secondary metabolites of plants and fungi. They cover a representative area of naturally occurring compounds and originate either from experiments or from simulations[2] based on the known structure of the compound. The database includes about 587 spectra with 3 to 35 peaks each, for a total of 7029. The peak distribution of all spectra in the database is shown in figure 5 left. The density in the peak space with all peaks in the database is shown in figure 5 right.

We generated larger data sets based on the real 2D-NMR data set while keeping the overall distribution of peaks. First, we estimated the mean of a Poisson distribution from the number of peaks per spectrum. The average number of peaks per spectrum is 12.3. The size of a newly generated spectrum is a random integer drawn from the Poisson distribution. In order to generate a new spectrum, we select a real spectrum at random and copy a fixed percentage p of

[2] ACD/2D NMR predictor, version 7.08, http://www.acdlabs.com/

Table 1: Examples for found duplicates in the real data

A.ID	A.Name	$\#peaks$	B.ID	B.Name	$\#peaks$
36	1R2_N_diethylnorephedrin	6	139	1R2S_N_diethylnorephedrin	6
37	1R2S_N_ethylephedrin	7	140	1R2S_N_ethylephedrin_wdh	7
38	1R2S_N_isopropylephedrin	7	141	1R2S_N_isopropylephedrin_wdh	7
72	Bicucullin	13	81	Capnoidin	13
99	Corynanthin	21	314	Yohimbin	20
168	Intermedin	12	194	Lycopsamin	12
248	BS_BC1_korr	5	349	BS_BC1_korr	5
277	Saframycin_A	15	279	Saframycin_S	15
302	Thalictrifolin	15	354	Cavidin	15
317	Flav3s	7	318	Flav3s4s	7
325	Flav573s	8	326	Flav573s4s	7
330	Flav73s4s	7	331	Flav74s	7
333	Isoflav3s	7	335	Isoflav4s	7
333	Isoflav3s	7	337	Isoflav53s	7
333	Isoflav3s	7	339	Isoflav54s	7
334	Isoflav3s4s	7	338	Isoflav53s4s	6
335	Isoflav4s	7	337	Isoflav53s	7
335	Isoflav4s	7	345	Isoflav73s	7
335	Isoflav4s	7	347	Isoflav74s	7
337	Isoflav53s	7	339	Isoflav54s	7
345	Isoflav73s	7	347	Isoflav74s	7

randomly selected peaks from that spectrum into the new one. This step is repeated until the spectrum has reached the predetermined size. In order to avoid identical peaks, Gaussian noise is added to the copied peaks. In the experiments we have set the percentage p to different values $10, 20, 40, 60$ without obtaining significant differences in the results. Therefore, we excluded this variable from our analysis.

6.2 Detection of Duplicates

First, we used the fuzzy duplicate definition to find duplicates in the real data. There were no duplicates intentionally included in the database. With a setting of $a = 3$ppm and $b = 0.3$ppm, which are reasonable tolerances, 54 of 171991 possible pairs are reported as fuzzy duplicates. A manual inspection revealed that 30 pairs are just very similar spectra, but 24 are candidates for real duplicates. Some pairs consist of an experimental and a simulated spectrum of the same substance, which confirms the usefulness of the definition. There was also a surprise, namely the pair Thalictrifoline/Cavidine. Both structures differ only in the stereochemical orientation of one methyl group. Evidently, in this case the commercial software package used in the simulation is unable to reflect the different stereochemistry in calculated spectra. In the future, fuzzy duplicates will be used to improve the quality of collections of 2D-NMR spectra. Some of the 54 duplicate pairs are presented in table 1.

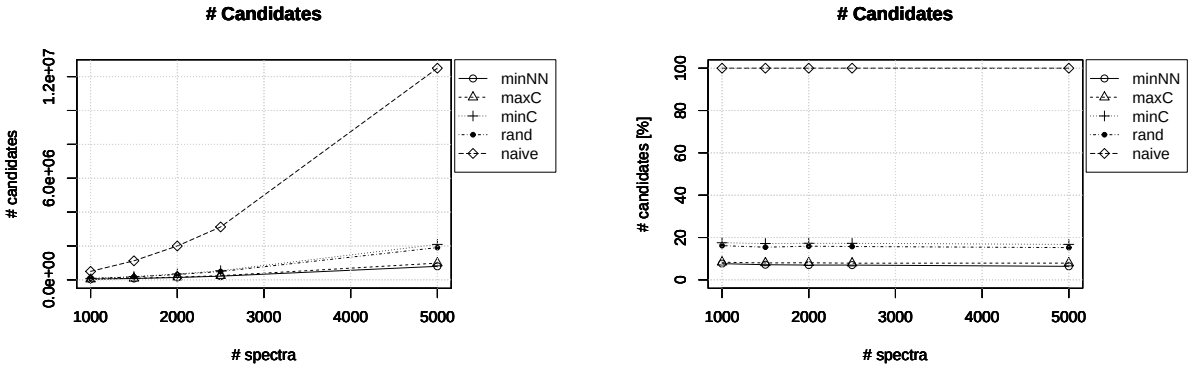

Figure 6: Number of candidate pairs versus number of spectra (left), percentage of candidate pairs versus number of spectra (right).

6.3 Performance Results

We conducted experiments with artificially generated data of sizes 1000, 1500, 2000, 2500, and 5000 spectra. All data sets include the set of real spectra as well as generated spectra. The set of duplicates was the same as for the real data in all cases, because it is unlikely that a duplicate is generated at random by our procedure.

First, we investigate how the number of candidate pairs found by a heuristic grows with the size of the data set. The results are shown in figure 6 (left). The naïve methods checks all $n(n-1)/2$ pairs, where n is the number of spectra. According to our argumentation in section 5.2 the optimal peak selecting heuristic minNN generates the smallest set of candidates. Because the shift-C and the shift-H coordinates are strongly correlated, the minC and minH as well as maxC and maxH respectively produce very similar results. Therefore, we present only the results for minC and maxC. The maxC heuristic produces only slightly more candidates. Random and minC are similar but have considerably more candidates than the optimal minNN heuristic. The better performance of maxC compared to minC and rand can be explained by the fact that the point density in the upper right corner of the peak space is lower than in the rest of the space (see figure 5 right). We normalized the number of candidates by the heuristics to be precentages of the number of pairs checked by the naïve methods, shown in figure 6 (right). The figure shows that the fractions stays constants as the data set grows. This indicates that the advantage of the heuristics will not degrade as the peak space becomes more and more densely populated with peaks.

In figure 7 we report the total run times of the heuristics. As the costs to find the candidates are different for the heuristics, we cannot expect that the situation for the plot with the number of candidates directly carries over to the run times. It shows that the optimal heuristic has a larger run time than the other three heuristics, because the costs to generate the candidates are much higher. The maxC shows the best run time results because the number of generated candidates is low and the costs to generate those candidate are comparable to minC and rand. So maxC with the given SQL implementation is the best choice to find duplicates in 2D-NMR data.

In future work we will investigate other classes of heuristics than single peak selection. Examples for such heuristics are (i) length of convex hull and (ii) size of bounding box. Both quantities vary little for fuzzy duplicates, because a pair of spectra can be a fuzzy duplicate

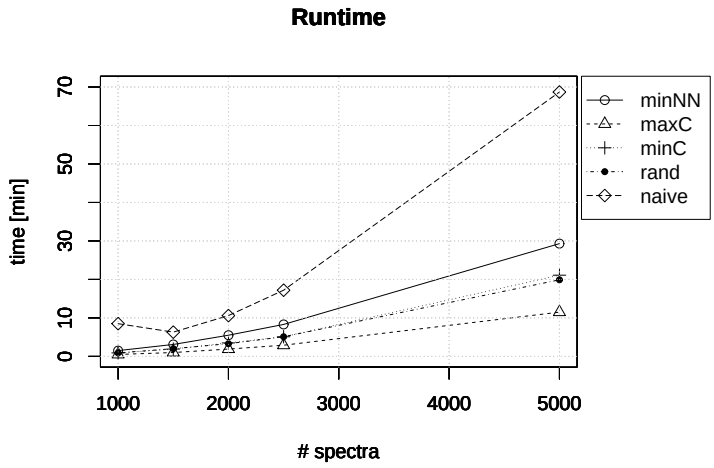

Figure 7: Total run time in minutes versus number of spectra.

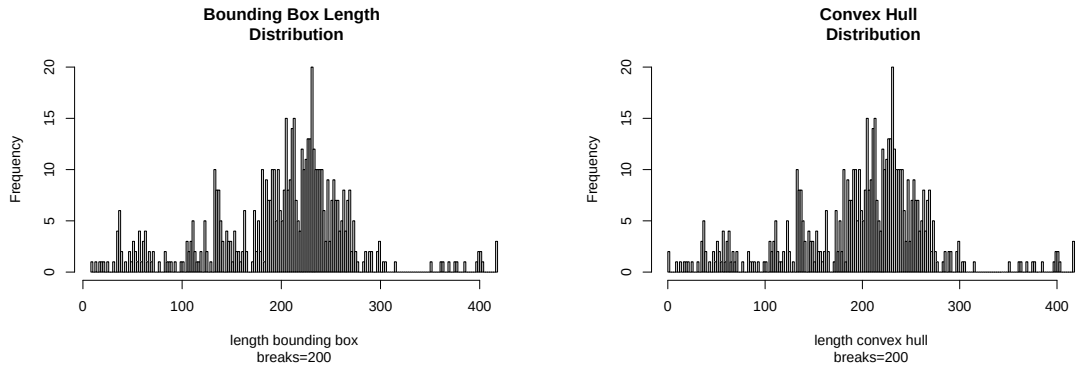

Figure 8: Histograms of the size of the bounding box (left) and the length of the convex hull (right) of a spectrum over all real spectra.

only if the difference between the lengths of the convex hulls are smaller than a threshold. The potentials of the heuristics are illustrated by the histograms in figure 8, which show that only a small fraction of spectra have similar lengths of the convex hull and sizes of the bounding box respectively. The plots are generated from the set of real spectra. However, it is an open question how to set the threshold to guarantee that no false negatives are produced.

7 Conclusion

We proposed a simple and robust definition for fuzzy duplicates of 2D-NMR spectra, which allows for measurement errors as well as merged peaks. We tested the definition on real data and gave a quadratic algorithm which can be implemented in SQL. We also proposed a class of heuristics, namely single peak selecting heuristics, which help to save runtime.

We gave the optimal heuristic of that class, which produces the least number of candidate pairs. However, in case the heuristics are implemented in SQL, the optimal heuristic has a larger runtime than simpler heuristics. This is due to the larger costs of the optimal heuristic to generate the candidates. A comparison with respect to run time revealed that the maxC

heuristic has the best tradeoff between costs for candidate generation and number of generated candidates. Future research will include the investigation of other classes of heuristics as well as mining procedures for frequent peak constellations, which are shared by many spectra.

References

[1] A. Tsipouras, J. Ondeyka, C. Dufresne et al. Using similarity searches over databases of estimated c-13 nmr spectra for structure identification of natural products. *Analytica Chimica Acta*, 316:161–171, 1995.

[2] A. S. Barros and D. N. Rutledge. Segmented principal component transform-principal component analysis. *Chemometrics & Intelligent Laboratory Systems*, 78:125–137, 2005.

[3] M. Bilenko and R. J. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 39–48, New York, NY, USA, 2003. ACM Press.

[4] A. Chowdhury, O. Frieder, D. Grossman, and M. C. McCabe. Collection statistics for fast duplicate document detection. *ACM Trans. Inf. Syst.*, 20(2):171–191, 2002.

[5] J. G. Conrad, X. S. Guo, and C. P. Schriber. Online duplicate document detection: signature reliability in a dynamic retrieval environment. In *CIKM '03: Proceedings of the twelfth international conference on Information and knowledge management*, pages 443–452, New York, NY, USA, 2003. ACM Press.

[6] F. Deng and D. Rafiei. Approximately detecting duplicates for streaming data using stable bloom filters. In *SIGMOD '06: Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 25–36, New York, NY, USA, 2006. ACM Press.

[7] D. Gomes, A. L. Santos, and M. J. Silva. Managing duplicates in a web archive. In *SAC '06: Proceedings of the 2006 ACM symposium on Applied computing*, pages 818–825, New York, NY, USA, 2006. ACM Press.

[8] M. Henzinger. Finding near-duplicate web pages: a large-scale evaluation of algorithms. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 284–291, New York, NY, USA, 2006. ACM Press.

[9] Y. Ke, R. Sukthankar, and L. Huston. An efficient parts-based near-duplicate and sub-image retrieval system. In *MULTIMEDIA '04: Proceedings of the 12th annual ACM international conference on Multimedia*, pages 869–876, New York, NY, USA, 2004. ACM Press.

[10] P. Krishnan, N. J. Kruger, and R. G. Ratcliffe. Metabolite fingerprinting and profiling in plants using nmr. *Journal of Experimental Botany*, 56:255–265, 2005.

[11] M. Farkas, J. Bendl, D. H. Welti et al. Similarity search for a h-1 nmr spectroscopic data base. *Analytica Chimica Acta*, 206:173–187, 1988.

[12] A. Metwally, D. Agrawal, and A. E. Abbadi. Duplicate detection in click streams. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 12–21, New York, NY, USA, 2005. ACM Press.

[13] G. N. Noren, R. Orre, and A. Bate. A hit-miss model for duplicate detection in the who drug safety database. In *KDD '05: Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 459–468, New York, NY, USA, 2005. ACM Press.

[14] D. W. Scott. Average shifted histograms: Effective nonparametric density estimators in several dimensions. *Ann. Statist.*, 13:1024–1040, 1985.

[15] C. Steinbeck, S. Krause, and S. Kuhn. Nmrshiftdb-constructing a free chemical information system with open-source components. *J. chem. inf. & comp. sci.*, 43:1733 –1739, 2003.

[16] M. Weis and F. Naumann. Detecting duplicate objects in xml documents. In *IQIS '04: Proceedings of the 2004 international workshop on Information quality in information systems*, pages 10–19, New York, NY, USA, 2004. ACM Press.

[17] H. Yang and J. Callan. Near-duplicate detection by instance-level constrained clustering. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 421–428, New York, NY, USA, 2006. ACM Press.