J Math Chem (2008) 44:75–87
DOI 10.1007/s10910-007-9294-y

ORIGINAL PAPER

An efficient algorithm for chemical fingerprinting

Hari Pulapaka · David R. Gibson

Received: 5 March 2007 / Accepted: 16 July 2007 / Published online: 21 September 2007

Abstract In this paper, we present and analyze a constructive polynomial-time algorithm that helps in solving the problem of comparing FT-ICR data to theoretical compounds. A brute-force approach to solving the problem is exponential in its worst-case. Our algorithm serves as the backbone of a more general interactive software tool for chemical fingerprinting which is to be developed in a subsequent paper. A preliminary algorithm analysis shows that our approach is efficient for most commercial applications involving the general problem of determining combinations of elements that satisfy a set of prescribed constraints.

Keywords FT-ICR · Algorithm · Polynomial · Fingerprinting · Data

1 Introduction

Fourier Transform Ion Cyclotron Resonance Mass Spectrometry (FT-ICR-MS or FT-ICR or just FT-MS) is widely considered the most versatile and complex method of mass analysis and detection. First developed in the early 1970s at the University of British Columbia by researchers Marshall and Comisarow [1], today it is one of the most sensitive methods of ion detection and enjoys a myriad of important applications in chemistry and the other basic sciences. FT-ICR depends heavily upon high speed

H. Pulapaka (✉)
Department of Mathematics and Computer Science, Stetson University, DeLand, FL 32723, USA
e-mail: hpulapak@stetson.edu

D. R. Gibson
Department of Mathematics and Computer Science, Valdosta State University, Valdosta, GA 31698, USA
e-mail: dgibson@valdosta.edu

computing and also the mathematical algorithms that determine the spectroscopic parameters from the numerical computer data.

In this paper, we provide a geometrically-based algorithm to help understand the chemical and biological relationships between a certain marine sediment and a specific oil sample. The marine sediment contains high levels of minerals ($CaCO_3$, $FeOOH$, silicates etc.), organic compounds (hydrocarbons, sulfides, etc.), bacteria, and other microbes (i.e., diatoms). The results in [2] include a preliminary study that aims to answer the following basic question: Is oil produced by massive underground bacterial colonies? Specifically, the authors compared a marine sediment sample which was extracted by an alcohol and analyzed by FT-ICR with the ICR data of an oil extract. Certainly, a conclusive analysis would be paramount in viewing oil as a renewable resource. Before we develop the notation for our paper, it would help to understand the techniques used in [2] so that the main problem that we have solved in this paper is in its true perspective. First, the two data sets were filtered producing a data set of only marine sediment compounds that had weight within a prescribed tolerance of a value from the oil data. This produced a data set with about 1,000 values. The idea was then to find matches in molecular weight between the filtered data set and a large number of known molecular compounds. In order to determine the matches, it was necessary to select a number of elements that might be present in the extracts and specify lower and upper bounds for each. For instance, in [2], the elements in the set $\{C, H, O, N, Na, S, Fe\}$ were ultimately chosen. Next, a computer program that compared the molecular weight of each sample with the total molecular weight of every valid combination of elements from the set above was developed. In each case, when the two values were within a prescribed tolerance (interval), the formula for the compound resulting from the chosen combination of elements was saved to a file for further analysis.

Implementing the process described above revealed a potential problem. The run time of the computer program could very easily become very long. For example, initially, 13 elements were specified along with the some reasonable lower and upper bounds. This resulted in having to compare each of the almost 38 billion potential compounds with the 1,000 real data points; a monumental task. Even after reducing the number of elements to only seven and a significant paring down of the bounds for each element, the comparison of the 104 million potential compounds with the real data points resulted in over 100 billion comparisons. In an effort to further simplify the analysis, the following constraint was imposed: the number of Hydrogen atoms must be less than or equal to twice the number of Carbon atoms plus two.

In general, in such analyses, it is useful to allow the researcher to specify tolerance (interval) constraints for the ratios of the number of atoms between pairs of elements. For instance, a researcher might be interested only in comparing compounds whose Nitrogen to Oxygen ratio is between 2 and 4. This further restricts the number of possible compounds that need to be compared with the real data.

As noted above, previous techniques to determine only the valid compounds subject to prescribed constraints required the generation of *all* possible compounds followed by the meticulous checking of each one to see which combinations satisfied the given constraints. The current project began as as effort to offer the user of the program an estimate of the expected run time in advance. However, we have now determined a constructive solution to the problem of determining which theoretical compounds would be the ones that would need to be compared with the real data. In other words, we will present an algorithm that actually generates all possible compounds that satisfy the Carbon–Hydrogen Rule and any ratio rules that may be provided.

2 Problem statement

We begin by defining the variables necessary to describe the problem and our solution. Let $E = \{x_1, x_2, \ldots, x_N\}$ denote the set of N elements chosen by the user. For each i, $1 \le i \le N$, let $L(x_i)$ and $U(x_i)$ denote the user-specified lower and upper bounds, respectively, for the number of atoms of element x_i and let $n(x_i)$ denote some integer value in this range. Next, recall that a user of the computer program can also specify that the ratios of the number of atoms of certain pairs of elements be fixed within a specified tolerance. We denote the number of such ratios defined by the user as R. For i and j where $1 \le i \ne j \le N$, let $r(x_i, x_j)$ denote the ratio of $n(x_j)$ to $n(x_i)$ with tolerance $t(x_i, x_j)$. That is to say, the user specifies that the following must be true

$$r(x_i, x_j) - t(x_i, x_j) \le \frac{n(x_j)}{n(x_i)} \le r(x_i, x_j) + t(x_i, x_j)$$

Now we are in position to clearly define the problem.
Problem Given $N, R, L(x_i), U(x_i), r(x_i, x_j), t(x_i, x_j)$, determine all the possible combinations of N elements that a user may choose from the set E and which satisfy each of the following constraints:

1. For each i, $1 \le i \le N$, we must have
$$0 \le L(x_i) \le n(x_i) \le U(x_i).$$
2. (Ratio Rule) For i and j where $1 \le i \ne j \le N$, we must have
$$r(x_i, x_j) - t(x_i, x_j) \le \frac{n(x_j)}{n(x_i)} \le r(x_i, x_j) + t(x_i, x_j).$$
3. $(C - H)$ Rule Denoting the elements carbon and hydrogen by C and H, respectively, if $n(C) \ne 0$, then we must have
$$n(H) \le 2n(C) + 2.$$

3 Developing an algorithm

We begin by determining the set of pairs of integers $(n(C), n(H))$ that satisfy the C–H Rule. Observe that this set consists of precisely all of the integer lattice points contained in the feasible region determined by constraint 1, (as it applies to carbon and hydrogen)

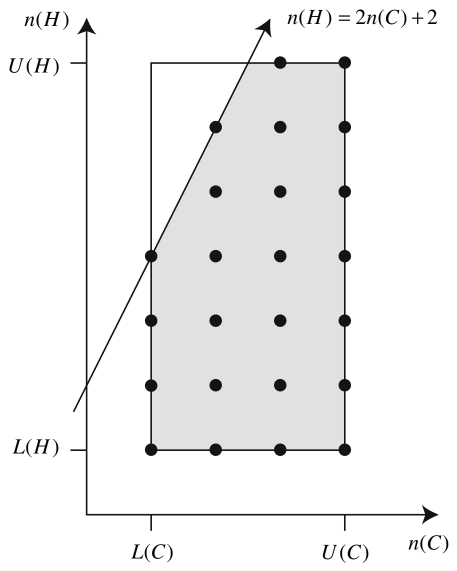

Fig. 1 Solutions to constraints 1 and 3

and constraint 3. Figure 1 gives a graphical view of such a feasible region. Observe that the solutions to constraint 1, consist of the integer pairs contained in the rectangular region in the first quadrant as shown in Fig. 1 while the solutions to constraint 3, consists of the half-plane below the line with slope 2 and y-intercept (0, 2). Thus, in order to list the set of integer pairs $(n(C), n(H))$ that satisfy constraints 1 and 3, we fix the first coordinate $n(C)$ and list the possible second coordinates $n(H)$ that will ensure that the pair $(n(C), n(H))$ lies on or *under* the lines given by $n(H) = U(H)$ and $n(H) = 2n(C) + 2$. It is easy to see then that the set $S(C, H)$ of carbon-hydrogen integer pairs that satisfy constraints 1 and 3 is given by

$$\begin{aligned} S(C, H) = \{&(L(C), L(H)), (L(C), L(H) + 1), \ldots, (L(C), \min(2L(C) + 2, U(H))),\\ &(L(C) + 1, L(H)), (L(C) + 1, L(H) + 1), \ldots,\\ &(L(C) + 1, \min(2(L(C) + 1) + 2, U(H))),\\ &(L(C) + 2, L(H)), (L(C) + 2, L(H) + 1), \ldots,\\ &(L(C) + 2, \min(2(L(C) + 2) + 2, U(H))),\\ &\vdots\\ &(U(C), L(H)), (U(C), L(H) + 1), \ldots, (U(C), \min(2U(C) + 2, U(H)))\}. \end{aligned}$$

Remark 1 Sometimes, it is possible that for a fixed value of $n(C)$ (say, $n(C) = s$), the corresponding second coordinates, $n(H)$, do not form an increasing sequence of integers. This occurs when the region below the line given by $n(H) = 2n(C) + 2$ does not intersect the rectangle given by constraint 1. Since we assume that $L(H) \leq U(H)$, this occurs precisely when $2s + 2 < L(H)$. In this case, we will adopt the convention of skipping that entire *row* in the set. That is to say, the set $S(C, H)$ will not contain any ordered pair with first coordinate equal to s.

We illustrate the set $S(C, H)$ with an example.

Example 1 Suppose $L(C) = 1$, $U(C) = 4$, $L(H) = 5$, and $U(H) = 10$. Then

$$\begin{aligned} S(C, H) = \{&(1, 5), (1, 6), \ldots, (1, \min(2(1) + 2, 10)),\\ &(2, 5), (2, 6), \ldots, (2, \min(2(2) + 2, 10)),\\ &(3, 5), (3, 6), \ldots, (3, \min(2(3) + 2, 10)),\\ &(4, 5), (4, 6), \ldots, (4, \min(2(4) + 2, 10)\} \end{aligned}$$

Note that the first row of the set above is invalid because $2(1) + 2 < 5$. That is to say, with $s = 1$, we have $2s + 2 < L(H)$. Hence, we delete the first row and determine that

$$\begin{aligned} S(C, H) = \{&(2, 5), (2, 6),\\ &(3, 5), (3, 6), (3, 7), (3, 8),\\ &(4, 5), (4, 6), (4, 7), (4, 8), (4, 9), (4, 10)\} \end{aligned}$$

Thus, of the 24 possible $(n(C), n(H))$ pairs (without regard to any constraints), only 12 satisfy the *C*–*H* Rule.

Next, as in the case of the C–H Rule, we determine the set of integer pairs that satisfy the ratio rules by studying the geometry of the feasible regions determined by constraints 1 and 2. Let x_i and x_j be any two elements for which the user has defined a ratio $r(x_i, x_j)$ and tolerance $t(x_i, x_j)$ requirement. Recall that $r(x_i, x_j) = \frac{n(x_j)}{n(x_i)}$. Again, constraint 1 defines a rectangular region in the first quadrant. For the sake of convenience of notation, let $a = a(i, j) = r(x_i, x_j) - t(x_i, x_j)$ and $b = b(i, j) = r(x_i, x_j) + t(x_i, x_j)$. Clearly, the equations given by $n(x_j) = an(x_i)$ and $n(x_j) = bn(x_i)$ denote straight lines through the origin with different slopes (otherwise, if $a = b$, then $r(x_i, x_j) = t(x_i, x_j) = 0$, which is impossible). In this case, the feasible region is given by the intersection of the regions below the line $n(x_j) = bn(x_i)$, above the line $n(x_j) = an(x_i)$, and inside the rectangle given by constraint 1 as it applies to the elements x_i and x_j (see Fig. 2). Similar to the C–H Rule, we fix $n(x_i)$ and list the possible values of $n(x_j)$. Thus, it is easy to see that the set of integer pairs $(n(x_i), n(x_j))$ that satisfy constraints 1 and 2 is given by

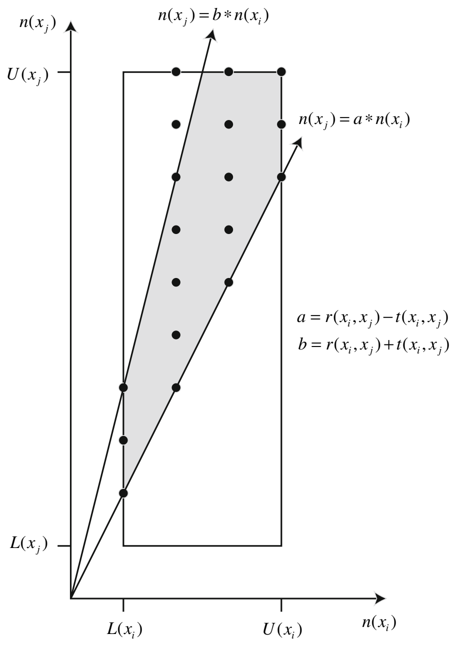

Fig. 2 Solutions to constraints 1 and 2 for a single pair of chosen elements

$$\begin{aligned}
&R(x_i, x_j)\\
&= \{(L(x_i), \max(\lceil aL(x_i)\rceil, L(x_j))), (L(x_i), \max(\lceil aL(x_i)\rceil, L(x_j)) + 1), \ldots,\\
&\quad (L(x_i), \min(\lfloor bL(x_i)\rfloor, U(x_j))), (L(x_i) + 1,\\
&\quad \max(\lceil a(L(x_i) + 1)\rceil, L(x_j))), (L(x_i) + 1, \max(\lceil a(L(x_i) + 1)\rceil, L(x_j)) + 1), \ldots,\\
&\quad (L(x_i) + 1, \min(\lfloor b(L(x_i) + 1)\rfloor, U(x_j))),\\
&\quad (L(x_i) + 2, \max(\lceil a(L(x_i) + 2)\rceil, L(x_j))), (L(x_i) + 2, \max(\lceil a(L(x_i) + 2)\rceil,\\
&\quad L(x_j)) + 1), \ldots, (L(x_i) + 2, \min(\lfloor b(L(x_i) + 2)\rfloor, U(x_j))),\\
&\quad \vdots\\
&\quad (U(x_i), \max(\lceil aU(x_i)\rceil, L(x_j))), (U(x_i), \max(\lceil aU(x_i)\rceil, L(x_j)) + 1), \ldots,\\
&\quad (U(x_i), \min(\lfloor bU(x_i)\rfloor, U(x_j)))\}.
\end{aligned}$$

Remark 2 Just as in the case of the C–H Rule, sometimes, it is possible to generate *invalid* integer pairs using the formula above. This occurs in one of two cases. Either the half-plane given by $n(x_j) > an(x_i)$ does not intersect the rectangle determined by constraint 1, or the half-plane given by $n(x_j) < bn(x_i)$ does not intersect the rectangle determined by constraint 1. That is to say, for some first coordinate (say, $n(x_i) = s$), we have either $as > \min(\lfloor bs\rfloor, U(x_j))$ or $bs < \max(\lceil as\rceil, L(x_j))$. In either case, we ignore that *row* in the set $R(x_i, x_j)$.

We illustrate the set $R(x_i, x_j)$ with an example that builds on Example 1.

Example 2 Suppose $x_i = C, x_j = H, L(C) = 1, U(C) = 4, L(H) = 5$, and $U(H) = 10$. Further suppose the user requires that the ratio of $n(H)$ to $n(C)$ be 1 with a tolerance of 1. That is to say, $r(C, H) = 1$ and $t(C, H) = 1$. In this case $a = r(C, H) - t(C, H) = 1 - 1 = 0$ and $b = r(C, H) + t(C, H) = 1 + 1 = 2$. Then

$$\begin{aligned}R(C, H) = \{&(1, \max(\lceil 0(1)\rceil, 5)), (1, \max(\lceil 0(1)\rceil, 5) + 1), \ldots, (1, \min(\lfloor 2(1)\rfloor, 10)),\\ &(2, \max(\lceil 0(1+1)\rceil, 5)), (2, \max(\lceil 0(1+1)\rceil, 5) + 1), \ldots,\\ &(2, \min(\lfloor 2(1+1)\rfloor, 10)),\\ &(3, \max(\lceil 0(1+2)\rceil, 5)), (3, \max(\lceil 0(1+2)\rceil, 5) + 1), \ldots,\\ &(3, \min(\lfloor 2(1+2)\rfloor, 10)),\\ &(4, \max(\lceil 0(1+3)\rceil, 5)), (4, \max(\lceil 0(1+3)\rceil, 5) + 1), \ldots,\\ &(4, \min(\lfloor 2(1+3)\rfloor, 10))\}\end{aligned}$$

Note that the first and second rows are invalid since with $s = 1$ and $s = 2$, respectively, we have $bs < \max(\lceil as\rceil, L(x_j))$. Consequently

$$R(C, H) = \{(3, 5), (3, 6), (4, 5), (4, 6), (4, 7), (4, 8)\}.$$

So, of the 24 possible $(n(C), n(H))$ pairs (without regard to any constraints), only six satisfy the Ratio Rule prescribed in Example 2.

Now, we are in a position to generalize the approach demonstrated above to the case where the user may choose more than two elements and specify several ratios between some or all of the chosen elements. We do so by computing all of the sets $R(x_i, x_j)$ and the set $S(C, H)$ and determining their *simultaneous intersection.*
We will conclude this section with an example that builds further on Examples 1 and 2 given above.

Example 3 Suppose the user chooses four elements $x_1 = C$ (Carbon), $x_2 = H$ (Hydrogen), $x_3 = P$ (Phosphorus), and $x_4 = O$ (Oxygen). Let the bounds on the four elements be as follows: $L(C) = 1, U(C) = 4, L(H) = 5, U(H) = 10, L(P) = 0, U(P) = 7, L(O) = 2$, and $U(O) = 6$. Next, suppose the user specifies the following ratios and tolerances: $r(C, H) = 1, t(C, H) = 1, r(C, P) = 2, t(C, P) = 0.5, r(O, H) = 1$, and $t(O, H) = 1$.
Note that we have already determined the sets $S(C, H)$ and $R(C, H)$ in Examples 1 and 2, respectively. Next, we must determine the sets $R(C, P)$ and $R(O, H)$. We do so exactly as demonstrated in Example 2.

In the case of $x_i = C$ and $x_j = P$, note that $a = 2 - 0.5 = 1.5$ and $b = 2 + 0.5 = 2.5$. Hence following the formula for the set $R(x_i, x_j)$ shown above Example 2, we have

$$\begin{aligned}R(C,P)=\{&(1,\max(\lceil 1.5(1)\rceil,0)),(1,\max(\lceil 1.5(1)\rceil,0)+1),\ldots,\\&(1,\min(\lfloor 2.5(1)\rfloor,7)),\\&(2,\max(\lceil 1.5(2)\rceil,0)),(2,\max(\lceil 1.5(2)\rceil,0)+1),\ldots,\\&(2,\min(\lfloor 2.5(2)\rfloor,7)),\\&(3,\max(\lceil 1.5(3)\rceil,0)),(3,\max(\lceil 1.5(3)\rceil,0)+1),\ldots,\\&(3,\min(\lfloor 2.5(3)\rfloor,7)),\\&(4,\max(\lceil 1.5(4)\rceil,0)),(4,\max(\lceil 1.5(4)\rceil,0)+1),\ldots,\\&(4,\min(\lfloor 2.5(4)\rfloor,7))\}\end{aligned}$$

Consequently,

$$R(C,P)=\{(1,2),(2,3),(2,4),(2,5),(3,5),(3,6),(3,7),(4,6),(4,7)\}.$$

Next, we determine the set $R(O,H)$. In this case, $a=1-1=0$ and $b=1+1=2$. Applying the formula for $R(x_i,x_j)$ with $x_i=O$ and $x_j=H$, we get

$$\begin{aligned}R(O,H)=\{&(2,\max(\lceil 0(2)\rceil,5)),(2,\max(\lceil 0(2)\rceil,5)+1),\ldots,(2,\min(\lfloor 2(2)\rfloor,10)),\\&(3,\max(\lceil 0(3)\rceil,5)),(3,\max(\lceil 0(3)\rceil,5)+1),\ldots,\\&(3,\min(\lfloor 2(3)\rfloor,10)),\\&(4,\max(\lceil 0(4)\rceil,5)),(4,\max(\lceil 0(4)\rceil,5)+1),\ldots,\\&(4,\min(\lfloor 2(4)\rfloor,10)),\\&(5,\max(\lceil 0(5)\rceil,5)),(5,\max(\lceil 0(5)\rceil,5)+1),\ldots,\\&(5,\min(\lfloor 2(5)\rfloor,10)),\\&(6,\max(\lceil 0(6)\rceil,5)),(6,\max(\lceil 0(6)\rceil,5)+1),\ldots,\\&(6,\min(\lfloor 2(6)\rfloor,10))\}\end{aligned}$$

Note that the first row is invalid since $bs<\max(\lfloor as\rfloor,L(x_j))$ with $s=1$. Consequently,

$$\begin{aligned}R(O,H)=\{&(3,5),(3,6),(4,5),\ldots,(4,8),\\&(5,5),\ldots,(5,10),(6,5),\ldots,(6,10)\}.\end{aligned}$$

Now, we have all the necessary information to determine the possible values for $n(C), n(H), n(P)$, and $n(O)$ that may be used to create valid combinations of the four elements C, H, P, and O. To do this, we have to determine the *intersection* of the various sets $R(x_i,x_j)$ along with the Carbon–Hydrogen Rule set $S(C,H)$. We do this by essentially inflating each set to take into account the elements that are not present in that set. This is done so that all of the rules involving a particular element are satisfied. We will complete this example in the next section after we formally present the algorithm that demonstrates this method and consequently, solves the problem.

4 The algorithm

In this section, we provide an algorithm to generate all of the valid combinations of elements chosen by the user subject to the user's constraints. Along with the notation used in previous sections, we will introduce some new notation during the course of developing the algorithm. Recall that N denotes the number of elements chosen by the user and R denotes the total number of ratio rules that need to be observed. Let $K = \max\{U(x_i) - L(x_i) \mid 1 \leq i \leq N\}$. Thus, K is the largest range provided by the user.

Our overall strategy is to construct a matrix of ordered pairs whose columns correspond to the elements in the analysis. Each ordered pair specifies a lower and upper bound for an element. A row in the matrix specifies a unique set of combinations of elements that satisfy constraints 1, 2, and 3. To do this, we first define a matrix R_{ij} for each constraint $R(x_i, x_j)$ whose rows satisfy that particular ratio constraint. Next, we define a similar matrix for the $S(C, H)$ constraint. Finally, we define a simultaneous intersection of all of these matrices to obtain the resultant matrix of solutions.

If we examine the definition of rows of the sets $R(x_i, x_j)$, we observe a symmetry that is useful in constructing an algorithm. For instance, we see that the first coordinate in each ordered pair in any particular row always has the same value. Noting this, we define

$$L_i = \min\{s \mid as \leq \min(\lfloor bs \rfloor, U(x_j)) \text{ and } bs \geq \min(\lceil as \rceil, L(x_j))\} \text{ and}$$
$$U_i = \max\{s \mid as \leq \min(\lfloor bs \rfloor, U(x_j)) \text{ and } bs \geq \min(\lceil as \rceil, L(x_j))\}$$

in order to determine the first and last valid rows of a set $R(x_i, x_j)$. In other words, we could say that the values of the first coordinate for any ordered pair in the first row must range from a lower bound of L_i to an upper bound of L_i. Likewise, the values of the first coordinate for any ordered pair in the second row ranges from $L_i + 1$ to $L_i + 1$, etc. We define a column vector of these lower and upper bounds for the values of the first coordinate for each row as follows

$$[(L_i, L_i)\ (L_i + 1, L_i + 1)\ \ldots\ (U_i, U_i)]^T$$

For each j, let r_{ij} denote the number of rows in any set $R(x_i, x_j)$. Thus, $r_{ij} = U_i - L_i + 1$. Considering the second coordinate of each ordered pair in a row, define

$$L_{jk} = \max(\lceil a(L_i + (k-1)) \rceil, L(x_j)), \quad k = 1, 2, \ldots, r_{ij} \text{ and}$$
$$U_{jk} = \min(\lfloor b(L_i + (k-1)) \rfloor, U(x_j)), \quad k = 1, 2, \ldots, r_{ij}.$$

Thus, we define a vector of lower and upper bounds for the second coordinates for each row as

$$[(L_{j1}, U_{j1})\,(L_{j2}, U_{j2})\ \ldots\ (L_{jr_{ij}}\ U_{jr_{ij}})]^T$$

Next, consider the *filler* vector

$$[(L(x_p), U(x_p))\ (L(x_p), U(x_p))\ \ldots\ (L(x_p), U(x_p))]^T \quad \text{where } p = 1, \ldots, N; \; p \neq i, j.$$

Now, we define a matrix R_{ij} for each set $R(x_i, x_j)$ that consists of a combination of the vectors defined above. In general, we define

$$R_{ij} = \begin{bmatrix} (L(x_1), U(x_1)) \ldots & (L_i, L_i) & \ldots & (L_{j1}, U_{j1}) & \ldots (L(x_N), U(x_N)) \\ (L(x_1), U(x_1)) \ldots & (L_{i+1}, L_{i+1}) & \ldots & (L_{j2}, U_{j2}) & \ldots (L(x_N), U(x_N)) \\ \vdots \quad \vdots & \vdots & \vdots & \vdots & \vdots \quad \vdots \\ (L(x_1), U(x_1)) \ldots & (U_i, U_i) & \ldots & (L_{jr_{ij}}, U_{jr_{ij}}) & \ldots (L(x_N), U(x_N)) \end{bmatrix}$$

We note that each row of R_{ij} specifies unique combinations of all elements that are valid with respect to the ratio of $n(x_j)$ to $n(x_i)$. We demonstrate the matrix R_{ij} with the parameters in Example 3. Suppose the elements are ordered: C, H, P, O. Then, using the definition for R_{ij} above, we have

$$R_{CP} = \begin{bmatrix} (1, 1) & (5, 10) & (2, 2) & (2, 6) \\ (2, 2) & (5, 10) & (3, 5) & (2, 6) \\ (3, 3) & (5, 10) & (5, 7) & (2, 6) \\ (4, 4) & (5, 10) & (6, 7) & (2, 6) \end{bmatrix}$$

The first row of R_{CP} specifies that all combinations where $1 \leq n(C) \leq 1, 5 \leq n(H) \leq 10$, $2 \leq n(P) \leq 2$, and $2 \leq n(O) \leq 6$ satisfy the ratio rule for Phosphorus to Carbon. Similarly, the remaining rows specify the remainder of the valid combinations.

Next, we obtain a corresponding matrix S_{CH} obtained from the set $S(C, H)$ in exactly the same way as we obtained the matrix R_{ij} with the following definitions

$$\begin{aligned} L_i &= L_C = \min\{s \mid 2s + 2 \geq L(H)\} \\ U_i &= U_C = U(C) \\ L_{jk} &= L_{Hk} = L(H), \quad \text{for } k = 1, 2, \ldots, r_{CH}. \\ U_{jk} &= U_{Hk} = \min(2(L_i + (k - 1)) + 2, U(H)), \quad \text{for } k = 1, 2, \ldots, r_{CH}. \end{aligned}$$

Finally, we define the intersection of all R of the ratio matrices R_{ij} and the matrix S_{CH} which yields a matrix that specifies all valid combinations. In other words, the intersection matrix specifies the solution to the original problem. First, define the intersection of a set of N ordered pairs as follows

$$(a_1, b_1) \cap (a_2, b_2) \cap \cdots \cap (a_N, b_N) = (\max(a_1, \ldots, a_N), \min(b_1, \ldots, b_N))$$

provided $\max(a_1, \ldots, a_N) \leq \min(b_1, \ldots, b_N)$; otherwise, we define the intersection to be *null*. Next, we define the intersection of a single row chosen from each matrix as follows

$$\begin{aligned}&[(a_1, b_1)\ \ldots\ (a_N, b_N)] \cap [(c_1, d_1)\ \ldots\ (c_N, d_N)] \cap [(p_1, q_1)\ \ldots\ (p_N, q_N)]\\ &\quad = [(a_1, b_1) \cap (c_1, d_1) \cap \cdots \cap (p_1, q_1)\ \ \ldots\ \ (a_N, b_N) \cap (c_N, d_N) \cap \cdots \cap (p_N, q_N)]\end{aligned}$$

A row is considered *null* if the pair-wise intersection of any of the ordered pairs is *null*. Finally, the resultant matrix is comprised of the non-null set of rows that correspond to the intersection of each row with every combination of rows from each matrix. We demonstrate this final construction by continuing and completing the solution corresponding to Example 3. Thus, the resultant matrix corresponding to the intersection $S(C, H) \cap R(C, H) \cap R(C, P) \cap R(O, H)$ is obtained as follows:

$$\begin{aligned}&\begin{bmatrix}(2,2) & (5,6) & (0,7) & (2,6)\\ (3,3) & (5,8) & (0,7) & (2,6)\\ (4,4) & (5,10) & (0,7) & (2,6)\end{bmatrix} \cap \begin{bmatrix}(3,3) & (5,6) & (0,7) & (2,6)\\ (4,4) & (5,8) & (0,7) & (2,6)\end{bmatrix}\\ &\cap \begin{bmatrix}(1,1) & (5,10) & (2,2) & (2,6)\\ (2,2) & (5,10) & (3,5) & (2,6)\\ (3,3) & (5,10) & (5,7) & (2,6)\\ (4,4) & (5,10) & (6,7) & (2,6)\end{bmatrix} \cap \begin{bmatrix}(1,4) & (5,6) & (0,7) & (3,3)\\ (1,4) & (5,8) & (0,7) & (4,4)\\ (1,4) & (5,10) & (0,7) & (5,5)\\ (1,4) & (5,10) & (0,7) & (6,6)\end{bmatrix}\\ &= \begin{bmatrix}(3,3) & (5,6) & (5,7) & (3,3)\\ (3,3) & (5,6) & (5,7) & (4,4)\\ (3,3) & (5,6) & (5,7) & (5,5)\\ (3,3) & (5,6) & (5,7) & (6,6)\\ (4,4) & (5,6) & (6,7) & (3,3)\\ (4,4) & (5,8) & (6,7) & (4,4)\\ (4,4) & (5,8) & (6,7) & (5,5)\\ (4,4) & (5,8) & (6,7) & (6,6)\end{bmatrix}\end{aligned}$$

We can see that the first four rows each specify $1 \times 2 \times 3 \times 1 = 6$ valid combinations, the fifth row specifies four combinations, and the last three rows each specify eight combinations for a total of 52 valid combinations of the four elements C, H, O, P. This completes the solution to the problem in Example 3.

5 Algorithm analysis

In this section, we present a theoretical and empirical analysis of our algorithm. We also provide a comparison with a brute force approach to solving the problem. We assume that the *primitive* operations (such as division, comparison etc.) are independent of the inputs. This is a valid assumption in general and certainly in the context of our specific application [3]. We will express the complexity as a function of the variables N and K defined in the previous section. In both cases, we will use the "big-oh" notation [4] to express the worst-case complexity.

First, we determine the worst-case complexity of a brute-force method. In this case, we first generate all possible combinations of N elements where each element has a maximum range of K. There are a maximum of K^N such combinations. Next, we note that the user may provide a maximum of $\frac{N(N-1)}{2}$ ratios between the N elements. In addition, we have the C–H Rule which adds one to the maximum number of rules that need to be followed. Thus, there are a total of $\frac{N(N-1)+2}{2}$ rules that need to be checked. Each rule involves either a comparison (C–H Rule) or a division (Ratio Rule) followed by a comparison. Thus, the worst-case complexity of such a method is at least $O(N^2K^N)$.

Next, we determine the worst-case complexity of the algorithm developed in the previous section. For each ratio set $R(x_i, x_j)$, the corresponding matrix R_{ij} has at most K rows and exactly N columns. And, there are at most $\frac{N(N-1)+2}{2}$ such matrices (Ratio Rules and C–H Rule). First, we determine the maximum *cost* of comparing any two matrices. Recall that each row of a matrix R_{ij} is compared (element-wise) with every row of the matrix $R_{(i+1)j}$. The comparison of any two rows requires N comparisons. Since there are at most K rows in every matrix, it takes at most KN comparisons between any given row of R_{ij} and all of the rows of $R_{(i+1)j}$. Thus it takes at most NK^2 comparisons between the matrices R_{ij} and $R_{(i+1)j}$. Finally, since there are at most $\frac{N(N-1)+2}{2}$ matrices, the worst-case complexity of our algorithm is $O(N^3K^2)$.

We conclude this section with a few remarks about the performance (theoretical and practical) of our algorithm. For fixed values of K and increasing values of N, our algorithm runs in polynomial time even in the worst-case whereas a brute-force method such as the one described above grows exponentially. It is easy to show that the only cases where the brute-force method may be faster than our algorithm are when $K = 1$ (which means every chosen element can take exactly one value) or $N \leq 4$. In all other cases, our algorithm is significantly more efficient than a brute-force method.

Certainly, in the context of our original problem of trying to study whether oil is produced by large underground colonies of bacteria, realistically, $N \leq 15$ and $K \leq 200$. For the sake of comparison, assuming a unit cost for all fundamental operations (divisions, comparisons etc.), when $N = 15$ and $K = 200$, our algorithm takes about 135 million fundamental operations whereas the brute-force method described takes nearly 7.37×10^{36} operations! This renders our algorithm readily implementable as opposed to one that is based on a brute-force method. In fact, in a subsequent paper [5], we provide an implementation and in-depth performance analysis of our algorithm.

In conclusion, we hope that the general nature of our algorithm and its implementation will provide researchers with a valuable and efficient tool to help analyze and determine the spectroscopic parameters from numerical computer data obtained through Fourier Transform Ion Cyclotron Resonance Mass Spectrometry. In fact, our algorithm may be easily adapted even to problems where all of the specified constraints are simply required to be monotonic (increasing or decreasing) and not necessarily just linear. Lastly, the worst-case complexity of our algorithm grows polynomially when the number of constraints increases polynomially. This is an invaluable characteristic that may be fully utilized when solving a wide variety of problems.

References

1. M.B. Comisarow, A.G. Marshall, Chem. Phys. Lett. **25** p. 282, (1974)
2. J. Bryant, T.J. Manning, D.R. Gibson, A.G. Marshall, R. Rogers, *Comparing Chemical Fingerprints: Is Oil the Product of Bacterial Production?* 58th Southeast Regional Meeting of the Americal Chemical Society, Augusta, GA, United States, November 1–4 (2006), SRM06-704. American Chemical Society, Washington, D.C. CODEN: 69INUY Conference; Meeting Abstract written in English. AN 2006:1191233 CAPLUS
3. T.H. Cormen, C.E. Leiserson, R.L. Rivest, *Introduction to Algorithms*. McGraw-Hill Publishing (1990)
4. C. Tovey, Interfaces (INFORMS) **32** (**3**), 30 (2002)
5. H. Pulapaka, D.R. Gibson, *A Fast Algorithm and Software for Analysis of FT-ICR Data*