Searching for a Single Mathematical Function to Address the Nonlinear Retention Time Shifts Problem in nanoLC-MS Data: A Fuzzy-Evolutionary Computational Proteomics Approach
Barton, Alan
Searching for a Single Mathematical Function to Address the Nonlinear Retention Time Shifts Problem in nanoLC-MS Data: A Fuzzy-Evolutionary Computational Proteomics Approach

Alan J. Barton

Abstract—Proteomics involves collecting and analyzing information about proteins within one or more complex samples in order to address a biological problem. One methodology is the use of high performance liquid chromatography coupled mass spectrometry (nanoLC-MS). In such a case, the accurate determination of non-linear peptide retention times between runs is expected to increase the number of identified peptides and hence, proteins. There are many approaches when using a computer for such a problem; including very interactive to completely non-interactive algorithms for finding global and local functions that may be either explicit or implicit.

This paper extends previous work and explores finding an explicit global function for which two stages are involved: i) computation of a set of candidate functions (results) by the algorithm, and ii) searching within the set for patterns of interest. For the first stage, three classes of approximating global functions are considered: Class 1 functions that have a completely unknown structure, Class 2 functions that have a tiny amount of domain knowledge incorporated, and Class 3 functions that have a small amount of domain knowledge incorporated. For the second stage, some issues with current similarity measures for mathematical expressions are discussed and a new measure is proposed. Preliminary experimental results with an EC algorithm called Gene Expression Programming (a variant of Genetic Programming) when used with a fuzzy membership within the fitness function are reported.

I. INTRODUCTION

A protein mixture from a biological sample is digested and the resulting peptides are injected into a mass spectrometer. This procedure may be performed for each sample of interest. For more detailed background information on mass spectrometry-based proteomics, see, for example, [1], [21], [9]. Fig. 1 shows three examples of sample vs. sample plots for retention time of peptides. The bottom row is a filtered version of the respective top row. The filter is based on the residual retention time. For example, if the residual is less than 200 scans then the points are relevant. 200 was arbitrarily selected in order to place the points of interest certainly within the filtered data. In addition, the motivation for filtering is to constrain the size and location of the search space. Example I in Fig. 1(Top,Left) shows that many data points are not of interest for modeling. In fact, there are 97,504 − 19,786 = 77,718 points that are clearly outliers. Example II in Fig. 1(Top,Middle) shows that most points are of interest for modeling. In this case, only 6,750 − 2,562 = 4,188 points are outliers. Example III in Fig. 1(Top,Right) shows that two regions of points of differing densities are present in such a way that the desired approximation model, $\mathcal{M}$, is located at the boundary of an upper high density region and a lower low density region. For this example, the fewest points are outliers 5,289 − 1,425 = 3,864.

Determination of a single mathematical function able to reconcile the nonlinear retention time shift problem within nanoLC-MS data is not trivial. This is due to both the complex nature of the biological samples investigated and the large number of observations (data points) that are not applicable (i.e. outliers). For the retention time problem, it is stated [15] that investigators must consider nonlinear shifts. The problem may be viewed from different perspectives; including prediction of retention time for one sample (e.g. [23]), or determination of retention times between two samples. For the former, [13] includes work on correcting a single sample based on amino acid retention time predictions. Some examples of computational approaches include the use of: i) linear regression [22] [15], ii) nonparametric regression (LOESS [6]) [15] [10], which has, among other good qualities, the following assumptions: a) errors are independently and normally distributed with constant variance, b) fitted function follows the pattern of the data (i.e. provides a nearly unbiased estimate), and c) the amount to smooth must be given as input to the algorithm. iii) wavelets [5], iv) variable penalty dynamic time warping [7], v) Genetic Algorithms [14], and vi) Cubic splines [16] among others.

A typical approach tends to use all data for model construction (e.g. linear regression). This is appropriate from the point of view that the obtained model is global in nature (i.e. one model for the whole data). However, some data points are not applicable for the construction of the unknown structure of the model. This paper focuses on addressing this issue through the use of: i) a fuzzy fitness measure, ii) an algorithm capable of constructing a mathematical expression, and iii) a population based algorithm in which sets of mathematical expressions are simultaneously considered. A parsimonious approach will be taken for the nonlinear retention time determination between two samples addressed within this paper. In particular, examples typified by Example II shown in Fig. 1(Bottom,Middle) will be considered. Tradeoffs to related work and between the proposed approaches are indicated throughout.
II. THEORY

To address the nonlinear retention time determination problem between two samples, a decision needs to be made whether an explicit analytical (e.g. [2]) or an implicit nonlinear function is desired. This paper investigates explicit analytical functions, which leads to the raising of a second question. Namely, should one global function be found or a set of local functions? The latter was investigated in [3]. The most appropriate set of conditions is not obvious. In addition, an algorithm’s search methodology may be oriented to search a space for a global or a local exact or approximate solution. The search may be guided by heuristics to regions of the search space that are more promising. The search methodology can also be based upon sets of solutions rather than only one solution. This paper proposes to use a global search to find sets of approximate global solutions.

A. Three Classes of Approximate Global Functions

From amongst the many possible global functions, this section presents three classes that may be appropriate for the nonlinear retention time determination problem.

1) Global Function Class 1: The global function \( f(x) \) is unknown and must be completely determined by a particular algorithm. For example, one possibility is to use an Evolutionary Computation (EC) algorithm, such as Genetic Programming for \( f(x) \)’s structure determination. Eq. 1 demonstrates this approach and shows that the algorithm will report \( g(x) \); it’s approximation to \( f(x) \). Since computers do not have infinite memory, a maximum size for \( g(x) \) may be specified a priori.

\[
f(x) = g(x)
\]  

2) Global Function Class 2: This class of global functions is similar to the first class, except that the data is rotated before the unknown function is determined by the algorithm Eq. 2. The motivation is that it may be the case that the algorithm can compute over the rotated data with greater ease.

\[
f(x) = \text{unrotate}(g(\text{rotate}(x)))
\]  

3) Global Function Class 3: This class of global functions is such that the data is rotated similar to class 2, but that the unknown function has a partially known form. In other words, the unknown function is parameterized by a set of functions that are determined by an algorithm. The motivation for this approach is that these function parameters may be simpler than the overall unknown function \( f(x) \), leading to the fact that the algorithm may find them more efficiently due to these constraints.

For example, this paper will consider the sinusoidal-like global function in Eq. 4.

\[
f(x) = a(x) \cdot \sin \left( \frac{2\pi x}{w(x)} \right)
\]  

The data within this paper is planar, therefore the following fact is stated. That is, a new planar co-ordinate \((x', f(x'))\) may be computed through a rotation angle \( \theta_r \) using the general formula in Eq. 3.

\[
x' = \left[ \frac{f(x)}{\sin(\tan^{-1}(\frac{f(x)}{x}))} \right] \cos \left[ \tan^{-1} \left( \frac{f(x)}{x} \right) + \theta_r \right]
\]  

\[
f\theta_r(x') = \left[ \frac{f(x)}{\sin(\tan^{-1}(\frac{f(x)}{x}))} \right] \sin \left[ \tan^{-1} \left( \frac{f(x)}{x} \right) + \theta_r \right]
\]  

Fig. 1. Three examples of retention time shifts. Top: raw nanoLC-MS scan vs. scan data. Bottom: Filtered version.
approximation, &\tilde{M}_i shown in Fig. 2(b). When comparing Fig. 2 to Fig. 1 it is concluded that this approach will require large amounts of computational power in order to determine $a(x)$ and $w(x)$ appropriately, since the constituent functions will need to be very malleable in order to cause the global function to flex into an appropriate shape. As such, a previous paper [3] investigated an approach focusing on the determination of a set of local models $\tilde{M}_i \text{deg}(.)$ for $i = 3$.

B. Discussion of Similarity between two Global Functions

Computation of a similarity between a mathematical expression $m_1$ and a second expression $m_2$ may be performed by computing a similarity based on i) their evaluations on a set of points or ii) the structure of their respective parse trees $p_1$ and $p_2$, i.e. $s_m(m_1, m_2) = s_p(p_1, p_2).$ This paper focuses on the latter, and $s_p(p_1, p_2)$ may be computed using a similarity measure for rooted, labeled trees. Such trees $T_i$ are a specific type of graph. For example, the following can all be computed [11]: i) the edit distance between two trees (for a review see [4]), ii) the size of their largest common sub-tree, iii) $\text{Prob}(T_2|T_1)$, the probability of receiving $T_2$ given that $T_1$ was transmitted across a channel causing independent substitution and deletion errors, and, iv) the a posteriori probability of $T_1$ being the transmitted tree given that $T_2$ is the received tree containing independent substitution, insertion and deletion errors. The following issues are considered for $s_m(m_i, m_j)$:

1) The cost associated to an operation (e.g. rotation, insertion, deletion, substitution, relabeling, etc.) within $p_i$ is not known and must be heuristically determined. For example, one possibility may be by evaluation of $m_i$ on a dataset of interest.

2) Modifications near the root of $p_i$ tend to cause much greater changes to the evaluation of $m_i$ than those near the leaves of $p_i$.

3) If nodes are not commutative (e.g. $\pm -$) then one approach would be to uniquely label them (e.g. $\pm _{\text{label}}$).

4) Multiple sets of operations, with differing costs, may lead to the same overall transformation from $p_i$ to $p_j$.

5) The $p_i$ are not ultrametric trees [8] (e.g. a dendogram is such a tree) because subtree evaluations are related through a node’s function, which may be arbitrary.

6) The largest common sub-tree, more generally called subgraph isomorphism, between $p_i$ and $p_j$ can be seen in Fig. 3. If the linear path "x—x" (Fig. 3(a), Fig. 3(b)) was changed to "x++x", then subgraph isomorphism could be used as a basis for constructing a similarity measure. This is because the associated expressions $m_i$ would then have the same evaluation. In addition Fig. 3(c) may be used as a starting point to construct an infinite set of $m_i$ that all have the common sub-tree "++x". Of course, similarity is then dependent on the subtrees $?_1$ and $?_2$. This indicates, for example, that multiple searches for common sub-trees would be required.

C. Proposed Mathematical Expression Similarity Measure

The details of the proposed method for similarity computation between mathematical expressions are shown in Alg. 1. The general idea is to construct a set of pairs $(r, u)$ of nodes for each tree ($r$ is the root node and $u$ is any node) and then
use this pairing information to calculate a similarity value. 

**Fig. 4** shows examples of pairs \((+, x)\) and how they may lie in relation to a matching pair \((+, x)\) within another tree. For a pair \((u, v)\) the pair separation distance \(d_s\) is calculated as an inverse relation to the number of edges lying on the path from \(u\) to \(v\); within **Fig. 4**, \(d_s((+, x)) = 1\).

---

**Algorithm 1: Mathematical Expression Similarity**

**Input**: Two mathematical expressions, \(m_1\) and \(m_2\)

**Output**: Computation of \(s_{m_1}(m_1, m_2) \in [0..1] \subset \mathbb{R}\)

1. Let \(p_1 \leftarrow \text{constructParseTree}(m_1)\);
2. Let \(p_2 \leftarrow \text{constructParseTree}(m_2)\);
3. Let \(r_1 \leftarrow \text{extractRoot}(p_1)\);
4. Let \(r_2 \leftarrow \text{extractRoot}(p_2)\);
5. Let \(\alpha = \text{heightOfTree}(p_1)\);
6. Let \(\beta = \text{heightOfTree}(p_2)\);
7. Let \(M_{p_1} = \{M^{p_1}_1, M^{p_1}_2, \ldots, M^{p_1}_s\} \leftarrow \text{enns}(r_1, r_1, 0)\);
8. Let \(M_{p_2} = \{M^{p_2}_1, M^{p_2}_2, \ldots, M^{p_2}_s\} \leftarrow \text{enns}(r_2, r_2, 0)\);
9. foreach node pair in \(M_{p_1}\) do
   10. | find first previously unmatched node pair in \(M_{p_2}\);
11. end
12. \(a_1 \leftarrow a_2 \leftarrow 0\);
13. \(b_1 \leftarrow \text{numberOfEdgesInTree}(p_1)\);
14. \(b_2 \leftarrow \text{numberOfEdgesInTree}(p_2)\);
15. foreach \(i \in \{1, 2\}\) do
   16. | foreach matched node pair in \(M_{p_i}\) do
   17. | \(d_s \leftarrow \text{distance for matched pairs (See Fig. 4)}:\)
   18. | \(a_i \leftarrow a_i + \frac{1}{3}\);
   19. | if \(d > 1\) then
   20. | \(b_i \leftarrow b_i + 1\);
   21. end
   22. end
23. end
24. \(s_{m_1}(m_1, m_2) \leftarrow s_p(p_1, p_2) \leftarrow \left(\frac{a_1}{b_1} + \frac{a_2}{b_2}\right) / 2\);

---

**Algorithm 2: Extract Node-Node Sets (enns)**

**Input**: Tree \(T\)'s root node \(r\), a current node \(c\) and a stack \(s\) containing the path from \(r\) to \(c\)

**Output**: An ordered collection of sets \(M_{T}^{d_s}\) of pairs \((r, u)\) where \(u\) is a node of \(T\) and \(d_s\) is the separation distance between \(r\) and \(u\)

1. push \((c\) onto \(s\);
   /* Pair the current node with all ancestors on path from root to it */
2. foreach \(u\in s\) do
   3. | insert \((u, c)\) into \(M_{T}^{d_s}\);
4. end
5. recurse on (possible children of \(c\));
6. pop \((s\):

---

D. Gene Expression Programming (GEP)

Biological evolution with its many complex interrelated processes [12] has inspired computational evolution. As such, a general Evolutionary Computation (EC) algorithm (Alg. 3) consists of a problem, \(P\), as input and a set of solutions, \(S\), as output. **Alg. 3** is composed of three stages: i) initialization **Line 1–4**, ii) processing generations **Line 5–11**, and iii) post-processing **Line 12–13**. The internal representation of a particular solution (\(S\)) and the reporting of that solution as the output of the algorithm (\(S\)) need not be the same. Such an explicit consideration of the algorithm’s results is of concern for practical reasons (e.g. [20]), but is not of direct concern within this paper.

Gene Expression Programming [17], [18] is a variant of Genetic Programming (GP). Each individual is an expression tree encoded as simple string of fixed length (called a chromosome) that represents entities of different sizes and shapes, generally nonlinear expressions (See **Alg. 4**). One feature of GEP is that the set of genetic operators applied to the chromosomes always produces valid expression [19].

1) **The Internal Representation**: The operators on the internal representation of an individual, \(I\), are used in order to construct new internal representations. This is called the *search methodology* and in **Alg. 3** refers to **Line 4,5,9,10**. Classical operators include selection, mutation, and crossing-over (or combining) of pieces from multiple individuals. The operators specific to GEP are listed in **Alg. 4**.

The internal representation of an individual, \(I\), for the
Algorithm 3: General Evolutionary Computation

Input: A problem, P. (i.e. a question)
Output: A set of solutions, S. (i.e. possible answer(s) to the question)
1 Let $\mathcal{A} \leftarrow \text{archivePopulations}(\emptyset, \emptyset)$;
2 Let $\mathcal{F}_P \leftarrow \text{createPopulations}(\emptyset)$;
3 computeFitness($\mathcal{F}_P$);
4 $\mathcal{A} \leftarrow \text{archivePopulations}(\mathcal{A}, \mathcal{F}_P)$;
5 while not terminationCriteriaSatisfied($\mathcal{F}_P$) do
6 \hspace{1em} Let $\mathcal{F}_{P'} \leftarrow \text{createPopulations}(\mathcal{F}_P)$;
7 \hspace{1em} computeFitness($\mathcal{F}_{P'}$);
8 \hspace{1em} $\mathcal{F}_P \leftarrow \text{combinePopulations}(\mathcal{F}_P, \mathcal{F}_{P'})$;
9 \hspace{1em} $\mathcal{A} \leftarrow \text{archivePopulations}(\mathcal{A}, \mathcal{F}_P)$;
10 \hspace{1em} $\mathcal{F}_P \leftarrow \text{selectIndividuals}(\mathcal{F}_P, \mathcal{A})$;
11 end
12 $S \leftarrow \text{selectIndividuals}(\emptyset, \mathcal{A})$;

Algorithm 4: Gene Expression Programming

Input: a dataset, D, and a fitness function for an individual, $I_{fit}$
Output: A set of $k$ most fit solutions, $S$
1 createInitialPopulationChromosomes();
2 expressChromosomes();
3 executeEachProgram();
4 evaluateFitness();
5 if terminate then
6 \hspace{1em} return $S$;
7 end
8 if not(bestOfGeneration) then
9 \hspace{1em} fitnessProportionateSelection();
10 begin Reproduction
11 \hspace{1em} replication();
12 \hspace{1em} mutation();
13 \hspace{1em} inversion();
14 \hspace{1em} IStransposition();
15 \hspace{1em} RISTransposition();
16 \hspace{1em} geneTransposition();
17 \hspace{1em} onePointRecombination();
18 \hspace{1em} twoPointRecombination();
19 \hspace{1em} geneRecombination();
20 end
21 newChromosomesOfNextGeneration;
22 goto Line 2;

nonlinear retention time determination problem is related to the specific global approximation case; Case I is in Eq. 6, Case II is in Eq. 7, and Case III is in Eq. 8.

\[ \tilde{M}_1 = \langle f(x) = \text{chromo}_1 \rangle \text{ See Eq. 1} \] (6)

\[ \tilde{M}_2 = \langle g(x) = \text{chromo}_1 \rangle \text{ See Eq. 2} \] (7)

The experiments used the settings reported in Table I. A large number of generations were used in order to allow the GEP algorithm an opportunity to search as much as possible within the constraints imposed by the parameter selections. The number of chromosomes within an individual is dependant upon the particular fitness function investigated. The addition, subtraction and multiplication functions were given twice as much weight as the other functions in order to bias the search towards simpler function choice by the

![Fig. 5. Retention time fuzzy membership function, $\mu_r$, for a peptide’s degree of belonging to an approximation model, $\tilde{M}$. See Eq. 11.](image_url)

\[ \tilde{M}_3 = \langle a(x) = \text{chromo}_1, w(x) = \text{chromo}_2 \rangle \text{ See Eq. 4} \] (8)

The complexity of $\tilde{M}_i$ is related to the complexity of $\tilde{M}_i$ with the property that they be as similar as possible.

2) The Fuzzy Fitness Function: The evolutionary search is oriented to minimizing or maximizing a fitness function. This function is computed, on an internal representation, by the algorithm. For the nonlinear retention time problem investigated, the fitness of an individual, $I_{fit}$, will be Eq. 9, which is based on fuzzy sets theory [24].

\[ I_{fit} = \text{fit} (\tilde{M}_i) \] (9)

where the $i$th approximation model’s fitness is Eq. 10

\[ \text{fit} (\tilde{M}_i) = \frac{\sum_{i=1}^{n} r \cdot (1 - \mu_r(r))}{n} \] (10)

with $n$ defined as the total number of points, $r = |\tilde{M}_i(x_{obs}) - y_{obs}|$ the model’s residual, and $\mu_r$ is a fuzzy membership function as Eq. 11. For the retention time determination problem within this paper $\mu_r$ has $v_1 = 1$, $v_2 = 5$ and is shown in Fig. 5.

\[ \mu_r(x) = \begin{cases} 1 & 0 \leq x \leq v_1 \\ \left( \frac{v_1 - v_2 - x}{v_1 - v_2} \right) & v_1 < x < v_2 \\ 0 & x \leq v_2; x < 0 \end{cases} \] (11)

The fuzzy fitness, $I_{fit}$, will have a value of zero if for the model, all points are close (i.e. $x \leq v_1 = 1$) to $\tilde{M}_i$. Further, $I_{fit}$ will be large when all points are far (i.e. $x \geq v_2 = 5$) from the model. The latter case is the mean residual, which is very similar to the root mean squared (RMS) error measure used within classical optimization.

III. EXPERIMENTAL SETTINGS

The experiments used the settings reported in Table I. A large number of generations were used in order to allow the GEP algorithm an opportunity to search as much as possible within the constraints imposed by the parameter selections. The number of chromosomes within an individual is dependant upon the particular fitness function investigated. The addition, subtraction and multiplication functions were given twice as much weight as the other functions in order to bias the search towards simpler function choice by the
TABLE I

<table>
<thead>
<tr>
<th>Fitness Function</th>
<th>Eq. 9 (See Eq. 1, Eq. 2, Eq. 4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimization Criterium</td>
<td>Maximize Fitness</td>
</tr>
<tr>
<td>Max No. Generations</td>
<td>1000</td>
</tr>
<tr>
<td>Number of Subpopulations</td>
<td>1</td>
</tr>
<tr>
<td>Size of the subpopulation</td>
<td>20</td>
</tr>
<tr>
<td>No. of Elite Individuals</td>
<td>1</td>
</tr>
<tr>
<td>Inversion</td>
<td>0.1</td>
</tr>
<tr>
<td>Mutation</td>
<td>0.044</td>
</tr>
<tr>
<td>IS Transposition</td>
<td>0.1</td>
</tr>
<tr>
<td>RIS Transposition</td>
<td>0.1</td>
</tr>
<tr>
<td>One Pt. Recomb.</td>
<td>0.3</td>
</tr>
<tr>
<td>Two Pt. Recomb.</td>
<td>0.3</td>
</tr>
<tr>
<td>Gene Recomb.</td>
<td>0.1</td>
</tr>
<tr>
<td>Gene Transposition</td>
<td>0.1</td>
</tr>
<tr>
<td>No. real constants/genome</td>
<td>2</td>
</tr>
<tr>
<td>Initialization Range</td>
<td>[0..1]</td>
</tr>
<tr>
<td>RAND Mutation</td>
<td>0.01</td>
</tr>
<tr>
<td>DC Mutation</td>
<td>0.044</td>
</tr>
<tr>
<td>DC Inversion</td>
<td>0.1</td>
</tr>
<tr>
<td>DC IS Transposition</td>
<td>0.1</td>
</tr>
<tr>
<td>No. Chromosomes</td>
<td>1 or 2</td>
</tr>
<tr>
<td>No. Genes</td>
<td>1</td>
</tr>
<tr>
<td>Gene Head size</td>
<td>8</td>
</tr>
<tr>
<td>Symbols</td>
<td>+, -, *,</td>
</tr>
<tr>
<td>Symbol Weights</td>
<td>2 for +, - , *</td>
</tr>
<tr>
<td>Random seed</td>
<td>4357</td>
</tr>
<tr>
<td>Number of Threads</td>
<td>1</td>
</tr>
</tbody>
</table>

TABLE II

<table>
<thead>
<tr>
<th>Fig.: Expression</th>
<th>4(a)</th>
<th>4(c)</th>
<th>4(e)</th>
<th>4(b)</th>
<th>4(d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4(a): x + x</td>
<td>0.375</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4(c): (x + 1) + x</td>
<td>0.375</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4(e): (x + 2) + (3 + x)</td>
<td>0.333</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4(b): (x + x) + 4</td>
<td>0.375</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4(d): x + (5 + x)</td>
<td>0.375</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4(f): (6 + x) + (x + 7)</td>
<td>0.333</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

IV. RESULTS

The two most fit candidate mathematical expressions are reported for each case and then the results are compared based on the proposed similarity measure after a discussion of the similarity measure’s properties with respect to two subgraph isomorphism based similarity measures.

A. Comparison to Subgraph Isomorphism Similarities

Let a subgraph isomorphism similarity measure \( s_{\text{sgi}} \) be naturally defined between two trees as in Eq. 12

\[
s_{\text{sgi}}(T_1, T_2) = \frac{\text{size(subgraph}(T_1, T_2))}{\max\text{size}(T_1, T_2)} \tag{12}
\]

and let tree \( T_1 \) be Fig. 3(a) and tree \( T_2 \) be Fig. 3(b). Then \( s_{\text{sgi}}(T_1, T_2) = 1 \) since the largest common subgraph is the whole graph; two terminal nodes hanging off one minus node attached to one minus node and then one terminal node. However, \( s_m(T_1, T_2) = 0.833 \) as computed via Alg. 1 indicates that the trees are not perfectly similar, which is the desired property for the proposed measure.

However, a more complicated variant of the measure in Eq. 12 could be defined, such as in Eq. 13.

\[
s_{\text{sgi}}(T_1, T_2) = \frac{\text{size(subgraph(unique}(T_1), unique(T_2))))}{\max\text{size}(T_1, T_2)} \tag{13}
\]

In this case, the nodes within the trees are considered special for some types of node. For example, the tree in Fig. 3(a) could have the minus nodes relabeled from \(-\) to \(-1\) and \(-2\). However, such a labeling should be performed carefully, since, for example, if the root of Fig. 3(a) was labeled \(-2\) and the root of Fig. 3(b) was labeled \(-1\) then the similarity would again be \(1\). This indicates that such a relabeling scheme must occur within the subgraph isomorphism similarity measure. If tree \( T_1 \) is Fig. 3(a) and tree \( T_2 \) is Fig. 3(b) then Eq. 13 evaluates to \( \frac{4}{5} = 0.8 \), which is less than the proposed measure’s value of \( s_m(T_1, T_2) = 0.833 \).

B. Ordering and Grouping of the Similarity Measure

The six families of mathematical expressions in Fig. 4 had their respective subtree replaced with constants in order to generate six concrete expressions. In particular, subtree \( ?_i \) was replaced with the subtree consisting of a single constant node \( i \) for \( i \in \{1..7\} \). For example, in Fig. 4(c) if \( ?_i \) is replaced by the subtree consisting of the constant \( 1 \) then the associated family member will be \((x + 1) + x\). The resulting pairwise expression similarities as computed via Alg. 1 are reported in the lower half of Table II.

For example, it is observed that \((x + x)\) is more similar to:

(i) \((x + 1) + x\),
(ii) \((x + x) + 4\), and
(iii) \((x + (5 + x))\) than to

(i) \((x + 2) + (3 + x)\) and
(ii) \((6 + x) + (x + 7)\), which is a desirable property for the proposed similarity measure.

C. Case I Candidates

Five selected candidate Case I solutions are Eq. 14, Eq. 15, Eq. 16, Eq. 17, and Eq. 18. All pairwise similarity computations for \( s_m(m_i, m_j) \) are also shown.

\[
0.985 + \left( x + \left( (x + x) + x \right)^{0.502} \right) \tag{14}
\]
\[
x + 0.985 \tag{15}
\]
\[
0.985 + x \tag{16}
\]
\[
x + 0.985 \tag{17}
\]
\[
0.985 - (0.502 - x) \tag{18}
\]

D. Case II Candidates

The two selected candidates are Eq. 19 and Eq. 20.

\[
\left( (x^{0.933} + x^{0.067}) + (x^{0.933} + 0.933) \right)^{0.933} \tag{19}
\]
\[
x^{0.933} \tag{20}
\]
E. Case III Candidates

Four selected candidate Case III solutions (for each of the two chromosomes) are Eq. 21, Eq. 22, Eq. 23, and Eq. 24.

\[
\begin{align*}
  c_1 &= 0.292 + 0.292 + e^{0.292} \cdot (x + 0.292) \quad (21) \\
  c_2 &= 0.297 + (0.673x - (0.673 + 0.673))^{(0.673+0.673)} \\
  c_1 &= 0.292 + 0.292 + e^{0.292} \cdot (x + 0.183) \quad (22) \\
  c_2 &= 0.297 + (0.673x - (0.673 + 0.673))^{(0.673+0.673)} \\
  c_1 &= 0.292 + (0.292 + (e^{0.183} + x + 0.292)) \quad (23) \\
  c_2 &= 0.297 + (0.673x - (0.673 + 0.673))^{(0.673+0.673)} \\
  c_1 &= e^{0.183} + x \quad (24) \\
  c_2 &= 0.297(0.673 + x)
\end{align*}
\]

F. Candidates Discussion

Overall, the candidate solutions are all linear in nature with differences lying in constants. For Case I, constants are used directly, for Case II, the constants are used as powers of \( x \), and for Case III, the constants are used in a much more complicated manner within exponentials. This is in agreement with the solutions reported in [3]. In particular, an evolutionary algorithm called Differential Evolution outperformed a nonparametric statistical technique called LOESS from the point of view of analytic solution closeness to desired data points. The current analytic results should be more deeply investigated, but they agree with the general linear class of function computed. This indicates the robustness of GEP to find radically different solutions when the fitness measure is changed. For the nonlinear retention time problem, further experiments are necessary in order to determine under what conditions the candidate solutions can be created that provide i) an adequate analytic solution, ii) a solution representation that is malleable enough in order to represent the investigated problem, and iii) a high probability of repeatability for good solutions.

V. CONCLUSIONS

An investigation of the problem of determination of retention time shifts between two nanoLC-MS data sets has been performed via a evolutionary computation algorithm for three classes of model. A similarity measure between pairs of mathematical expressions was also introduced with promising results reported. The results were also compared to other approaches that both did and did not use the fuzzy membership function. Overall, this paper indicates the preliminary nature of the approach and the degree to which further experimentation is necessary, including a methodology for further automation and an investigation of other fitness measures, such as ones based on Rough Sets theory.

REFERENCES