The Journal of Supercomputing (2020) 76:9168–9188
https://doi.org/10.1007/s11227-020-03206-0

NestMSA: a new multiple sequence alignment algorithm

Mohammed Kayed[1] · Ahmed A. Elngar[1]

Published online: 19 February 2020

Abstract
Multiple sequence alignment (MSA) is a core problem in many applications. Various optimization algorithms such as genetic algorithm and particle swarm optimization (PSO) have been used to solve this problem, where all of them are adapted to work in the bioinformatics domain. This paper defines the MSA problem, suggests a novel MSA algorithm called 'NestMSA' and evaluates it in two domains: Web data extraction and removing different URLs with similar text (DUST). The suggested algorithm is inspired by the PSO optimization algorithm. It is not a generalization of a two-sequence alignment algorithm as it processes all the sequences at the same time. Therefore, it looks globally at the same time on all sequences. Different from other PSO-based alignment algorithms, swarm particles in the proposed NestMSA algorithm are nested inside the sequences and communicated together to align them. Therefore, global maximum is guaranteed in our algorithm. Furthermore, this work suggests a new objective function which both maximizes the number of matched characters and minimizes the number of gaps inserted in the sequences. The running time complexity and the efficiency of NestMSA are addressed in this paper. The experiments show an encouraging result as it outperforms the two approaches DCA and TEX in the Web data extraction domain (95% and 96% of recall and precision, respectively). Furthermore, it gives a high-performance result in the DUST domain (95%, 93% and 92% of recall, precision and SPS score, respectively).

Keywords Web mining · Particle swarm optimization · Sequence alignment

✉ Ahmed A. Elngar
elngar_@yahoo.co.uk

Mohammed Kayed
mskayed@gmail.com

1 Faculty of Computers and Artificial Intelligence, Beni-Suef University, Beni-Suef, Egypt

1 Introduction

Sequence (string) alignment is a common and a core problem in many fields such as information extraction, removing DUST, bioinformatics, NLP, financial data analysis and others. Aligning of characters in multiple data sequences is a challenge because these characters have many variations. The first variation is called missed characters/patterns, in which a character (a pattern of different consecutive characters) may be missed (has no occurrences) in a sequence. Gaps (referred by—in this paper) are inserted among the sequence characters in place of these missed characters. In the second variation, called repetitive characters/patterns, a character (a pattern) may have more than one occurrence in a sequence. The third variation is called multi-order characters/patterns. Characters (patterns) in this variation may have multiple ordering in different sequences. The fourth and last variation is the existence of what are called disjunctive (mutative) characters, in which a character (a pattern) in a sequence may appear as an alternative to another character (pattern) in some other different sequences. Multiple sequence alignment aims to arrange sequences of characters in order to recognize missed (gaps), repetitive, multi-order and disjunctive (mutative) characters.

Many research algorithms have been proposed to solve the MSA problem, although it is still of great necessity for different reasons. First, some proposed algorithms solve the multiple sequence alignment problem as a generalization of the pair-wise alignment problem (i.e., align the sequences two by two until finally cover the whole sequences). The results of these pair-wise alignment methods usually are affected by the order of the sequences to be aligned and will not be applicable to unseen sequences. Second, many recent MSA algorithms use optimization models such as particle swarm optimization (PSO), although global maximum alignment is not guaranteed as they consider the whole sequences as a solution (a particle) and then try to modify each particle (inserting gaps in the sequences) step by step until reaching a best local solution of this particle. The global best solution is then gained by tracking the local best solutions for all working particles. Third, to the best of our knowledge, all of these algorithms are adapted to work in the bioinformatics domain (e.g., DNA, RNA or Protein alignment) [1], although MSA is a core for other domains such as Web data extraction and removing DUST. Thus, MSA is still demanded to be addressed in such domains. In the field of Web data extraction, data to be extracted are embedded in the Document Object Model (DOM) tree of a Web page. Tree alignment is used to solve the data extraction problem (Depta [2] and FiVaTech [3]). For simplicity, tree alignment is usually solved by converting the multiple tree alignment problem into the multiple sequence alignment. In the field of detecting and removing duplicate Web pages without fetching their contents (DUST), Web pages URLs get tokenized into tokens and then are aligned to generate a set of normalization rules (regular expressions). Later, these expressions are used by search engines to avoid crawling duplicate URLs [4, 5].

In this paper, the multiple sequence alignment problem is defined and a new alignment algorithm called 'NestMSA' is proposed to solve this problem. Our proposed algorithm tries to handle the above-mentioned variations: missed patterns, repetitive patterns, multi-order patterns and disjunctive patterns. Similar to the peer matrix alignment algorithm (a heuristic-based one) in [6], our proposed algorithm processes multiple sequences and aligns all of them at the same time. It is not a generalization of a two-sequence alignment algorithm. Therefore, NestMSA has a global view for the whole sequences to be aligned. NestMSA is inspired by the particle swarm optimization algorithm. Different from other PSO-based MSA algorithms, swarm particles in the proposed algorithm are nested/reside inside the sequences and they are communicated together to align these sequences step by step. In other words, a particle in NestMSA corresponds to a character inside the sequences, while a particle in other PSO-based alignment algorithms corresponds to the whole sequences to be aligned. Another contribution in this paper is that a new objective function which tries to both maximize the number of matched characters and minimize the number of gaps inserted is provided. Therefore, global maximum is guaranteed in the proposed algorithm. Furthermore, different from most optimization-based MSA algorithms, the proposed algorithm will be evaluated and addressed in the two different domains: Web data extraction and removing DUST.

The rest of the paper is organized as follows. Section 2 presents related works. Section 3 introduces the particle swarm optimization algorithm, while the MSA problem is formulated in Sect. 4. The proposed PSO-based algorithm with a simple example is presented in Sect. 5. The experimental study is described in Sect. 6. Finally, our work is concluded in Sect. 7.

2 Related works

Multiple sequence alignment has been solved by using many approaches in different fields [7–9]. In the bioinformatics domain, there are four main different approaches. The first approach is an exact approach in which dynamic programming is used to align two sequences. It converts the MSA problem into the problem of finding the shortest path in a weighted direct acyclic graph. Dynamic programming gives good alignment results for two sequences. When extended to multiple sequence alignment, the algorithm showed poor results as the lengths and the number of sequences increased. In the second approach, called a progressive approach, the algorithm first finds the most similar sequences and then incrementally aligns the other sequences to the initial alignment. An example of this approach is CLUSTAL X (an updated version of CLUSTAL W, which is also an extension of a previous version CLUSTAL) [10]. The third approach is a consistency-based approach. It considers that an optimum alignment occurred when optimal pair-wise alignments have been reached. A common example is T-COFFEE [11]. It uses a library of pair-wise alignments from the sequences to be aligned, then it finds both the best global and the local alignments. After that, it gets a final alignment which has the highest level of consistency within the library. The last approach is called an iterative approach. The alignment algorithm in the iterative approach begins with a generated initial alignment, and then it refines the alignment until no more improvement could be obtained. GA and

PSO are the most common algorithms used for this iterative approach. Our proposed algorithm is an iterative PSO-based method.

Many approaches in the bioinformatics domain adapt PSO algorithm to solve the MSA problem in which the objective is to maximize a scoring function among the sequences to be aligned. Examples of such approaches are [12–14]. They consider a particle in the swarm as a set of multiple sequences to be aligned. A particle is represented as a set of vectors, where each vector holds positions of the gaps in a sequence to be aligned. The scoring function used to keep track of the global best particle position is based on the alignment score of each pair of sequences. The alignment score between two sequences is the summation of the scores assigned to match each pair of symbols in these two sequences. The only difference among these approaches is the way that the particles are initialized. The former approach initializes the particles randomly, while the latter one generates the initial particles using the solution obtained from ClustalX algorithm [10]. There are two main differences between such PSO-based algorithms and our proposed algorithm. First, to the best of our knowledge, all PSO-based algorithms are adapted and addressed in the bioinformatics domain. Second, a particle in all of the PSO-based algorithms corresponds to a whole peer matrix with gaps added among the sequences, while a particle in our proposed algorithm corresponds to just a symbol in a matrix row as we shall discuss later.

MSA is not addressed and tested in the Web data extraction domain, although most of the proposed approaches in this domain are solved by using MSA. Some of these approaches are OLERA [15], RoadRunner [16], DEPTA [2], VIPER [17], FiVaTech [3], AFIS [18], TEX [19] and DCA [20]. Wrappers (extraction rules) that are used to extract data from Web pages in OLERA are constructed by acquiring a rough example from the user. It applied the two main techniques: pattern mining and string alignment. OLERA considered the problem of multiple string alignment as a generalization of two-string alignment. RoadRunner assumed that the input pages are similar and each one is a sequence of tokens. Extraction rules are recognized by aligning two pages (sequences), and then the alignment result is compared with a third page (sequence). DEPTA identified data record boundaries by using string edit distance. After that, attributes in these records are identified using partial tree alignment. ViPER recognized and ranked data records with respect to the users visual perception of the input Web page. It handled missed and multi-valued data by using the edit distance after applying the tandem repeats algorithm. String alignment in VIPER is based on the global matching and the text content. To simplify the multiple alignment problem, it used a divide-and-conquer approach. AFIS formulated the problem of data extraction as alignment of Document Object Model (DOM) Trees leaf nodes. AFIS is an unsupervised approach which deduces schema of detail pages that include all product information to be extracted. It applied divide-and-conquer and longest increasing sequence (LIS) algorithms to mine landmarks from input. Finally, FiVaTech is an unsupervised page-level data extraction approach. It applied the three techniques: tree matching, tree alignment and pattern mining, to solve the data extraction problem. First, FiVaTech merged all input DOM trees into a single tree (pattern tree), which is then used to detect the template and the schema of the site. To merge multiple trees, the tree alignment problem is converted into the multiple string alignment problem. FiVaTech is used in this work to collect a dataset which consists of peer matrixes to be aligned.

To the best of our knowledge, the only two MSA algorithms that are suggested and addressed in the field of Web data extraction are TEX and DCA (that is why both have been chosen in our experiment to be compared with our proposed one). Although the two methods use MSA, the methodologies of the two algorithms are different. TEX is an unsupervised information extraction approach that tried to identify a Web site template (shared patterns) and then removed this template from the input site pages. Different from DCA and FiVaTech, TEX did not use the pages DOM trees, rather it considered each input page as a sequence of tokens. Given a set of multiple sequences/pages $S = \{T_1, T_2, \dots, T_n\}$, where $n \geq 2$ as input, TEX tried to separate the template used to generate these pages from the data to be extracted. In order to achieve that, TEX used a simple multiple string alignment which is based on a searching algorithm that looks for shared patterns. It iteratively searched for a shared pattern of sizes $\max, \max - 1, \dots, \min$, in order (min and max values are calculated experimentally). If a shared pattern is found, TEX expanded the set S by replacing each sequence T_i with three other sequences: prefix, separator and suffix of the detected pattern in the sequence T_i, and so on until no shared patterns are found.

In DCA [20], Oviliani and Chang emphasized the necessity to have a high-performance multiple string alignment algorithm to solve the Web data extraction problem (from singleton Web pages). Given more than one DOM tree as input, each sequence of leaf nodes in a DOM tree is added as a row in a matrix which is finally aligned in two phases: a template mining phase and an alignment phase. Leaf nodes are classified into one of the three categories: mandatory template, optional template and mandatory data. The first two, parts of the Web site template, are used as landmarks to achieve the alignment process, while the last one corresponds to the data to be extracted. In the template mining phase, a divide-and-conquer approach is applied to recursively detect landmarks (mandatory templates) via longest increasing sequence (LIS). Further, optional templates are detected and merged across segments to remove false-positive mandatory templates. In the alignment phase, non-template leaf nodes are aligned to generate a consistent output for multi-order attribute-value pairs and merge similar/disjunctive columns to generate the output matrix.

Detecting and removing DUST is another domain which is recently solved by MSA algorithms [4]. Rodrigues et al. have suggested an approach called DUSTER, which aligns multiple URLs with duplicated content. It then uses the alignment results to construct high-quality, general and precise normalization rules. The derived rules could be used by a web crawler to avoid fetching DUST. To align a set of multiple URLs, they used the progressive alignment strategy which is a generalization of pair-wise sequence alignment.

3 Particles swarm optimization

Given a fitness function that describes a problem to be optimized, classical optimization algorithms use the first derivative of this fitness function to get the optima on a given constrained surface. Recently, many optimization algorithms have been suggested to solve the problem without calculating the derivative which is very difficult, especially for discontinuous optimization spaces. Examples of such algorithms are: GA [21, 22] and PSO [23–25]. Particle swarm optimization is a stochastic search algorithm and an alternative solution to the nonlinear optimization problem. The algorithm has been inspired by the social behavior of animals such as bird flocking, fish schooling and so on. It was introduced in 1995 by Kennedy and Eberhart. In PSO, the population (swarm) consists of a set of particles (solutions). The PSO algorithm works as follows. Initially, particles in the swarm are distributed randomly in the search space in such a way that each particle $p_i; (1 \leq i \leq m)$ has an initial position $x_i(t_\circ)$ and moves toward a randomly chosen direction. Each particle keeps track of its best previous position $\Im_i$ and a global best position g which is shared by all other particles. After each particle has moved to a new position, iteratively it improves its new position by flying toward both its best position and the global best position. PSO will be terminated when the maximum number of iterations (calculated based on some stop criteria) has been reached. Considering maximization problems, the particle best position and the global best position are calculated based on an objective (fitness) function $f(x); x \in R^N$; where N is the dimension of the search space. The local best position $\Im_i$ of the particle p_i and the global best g of all particles at the iteration number $t + 1$ are defined, respectively, as:

$$\Im_i(t+1) = \begin{cases} \Im_i(t), & \text{if } f(x_i(t+1)) \leq f(\Im_i(t)) \\ x_i(t+1), & \text{if } f(x_i(t+1)) > f(\Im_i(t)) \end{cases} \tag{1}$$

$$g(t+1) = \Im_i^*(t+1); \text{ such that } f(\Im_i^*(t+1)) = \max_{\{i=1,\ldots,m\}} \{\Im_i(t+1)\} \text{ and } m \geq 1 \tag{2}$$

The position x_i of the particle p_i is updated using the following formula:

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{3}$$

The velocity v_i of the particle p_i at iteration $t + 1$ is calculated by:

$$v_i(t+1) = \omega v_i(t) + c_1 r_1(t)[\Im_i^*(t) - x_i(t)] + c_2 r_2(t)[g(t) - x_i(t)] \tag{4}$$

where

(i) The term $\omega v_i(t)$ is called an inertia component which prevents drastic change in the direction of the particle.
(ii) The term $c_1 r_1(t)[\Im_i^*(t)$ is called a cognitive component which directs the particle to fly toward its previous best position.
(iii) The term $c_2 r_2(t)[g(t) - x_i(t)]$ is called a social component which directs the particle to fly toward the best position found by the particles neighborhood (the global best).
(iv) The constant ω is called an inertia weight.
(v) The two constants c_1 and c_2 are positive constants that accelerate the cognitive component and the social component, respectively.

4 MSA problem formulation

Given a finite set A of arbitrary symbols (called an Alphabet set), a sequence S_i of length n_i is defined as a series of ordered symbols taken from the alphabet A (i.e., $S_i = \{s_{1i}, s_{2i}, \ldots, s_{n_i i}\}$, where $n_i = |S_i|$. The jth symbol in the sequence S_i is denoted by s_{ji} ($j = 1,\ ,n_i$). We define the term multi-sequence matrix or simply called peer matrix as follows.

Definition 1 (*Peer matrix*) Given a set of c sequences $T = S_1, S_2, \ldots, S_c$ taken from the alphabet A, where the sequence S_i has a length $n_i (i = 1, \ldots, c)$, a peer matrix M is defined as the matrix whose columns are the sequences $S_1, S_2, \ldots, S_c$.

That is:

$$M = \begin{bmatrix} s_{11} & s_{12} & \cdots & s_{1c} \\ s_{21} & s_{22} & \cdots & s_{2c} \\ \cdots & \cdots & \cdots & \cdots \\ s_{n_1 1} & s_{n_2 2} & \cdots & s_{n_c c} \end{bmatrix}$$

Multiple sequence alignment is now defined as follows.

Definition 2 (*Multiple sequence alignment*) Given a peer matrix M formed from multiple sequences taken from the alphabet set A, multiple sequence alignment is the problem of finding a matrix M' (called an aligned matrix) from the original matrix M such that:

$$M' = \begin{bmatrix} s'_{11} & s'_{12} & \cdots & s'_{1c} \\ s'_{21} & s'_{22} & \cdots & s'_{2c} \\ \cdots & \cdots & \cdots & \cdots \\ s'_{n_1 1} & s'_{n_2 2} & \cdots & s'_{n_c c} \end{bmatrix}$$

where M' is taken from a new alphabet $A' = A \bigcup \{—\}$ and it has all of the following constraints:

1. The sequences (columns) $S'_1, S'_2, \ldots, S'_c$ of M' are defined over the new alphabet A'.
2. The sequences $S'_1, S'_2, \ldots, S'_c$ of M' have the same length $n = \max\{|S'_i|; i = 1, \ldots, c\}$.
3. $S'_i (i = 1, \ldots, c)$ reverts to the original S_i by removing all the inserted gap characters.
4. For each row r in the matrix M', there exists at least one symbol s'_{ri} which is not a gap; i.e., $s'_{ri} \neq$ '—.' This means that it is not possible to have a row in the matrix M' that only has gaps.
5. Number of matched characters (symbols) in each row of M' is maximized.
6. Number of gaps inserted in the matrix M' is minimized.

In the multiple sequence alignment of DNA sequences, the alphabet A is A, C, T, G. Similarly, the RNA alphabet is A, C, U, G, while the alphabet for protein MSA is $A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y$. In the NLP domain, the alphabet is a set of all tokens/words defined by the language. In the domain of detecting and removing DUST, the alphabet is a set of URL tokens that have been described by the EBNF-based grammar [4] (e.g., the URL http://domain/story?id=num is tokenized to have the following 11 tokens $\{<http>, < : >, </>, </>, <domain>, </>, <story>, <?>, <id>, < = >, <num>.\}$ Finally, for the Web data extraction problem, the alphabet is a set of identifiers (symbols) where each one corresponds to a subtree (a leaf node) from a Web page DOM tree that includes some data to be extracted. Matched/similar subtrees (leaf nodes) take the same identifier (symbol) where similarities could be measured by using tree-edit distance, visual features comparison, or any other measurement way. This means, for the last two domains, removing DUST and Web data extraction, the peer matrix contains sequences of symbols/characters that correspond to URL tokens or subtrees from the DOM tree, respectively.

Figure 2a shows a simple example of four sequences alignment problem. The matrix to be aligned M consists of the four sequences: $S_1 = \{a, b, c, b, c, d, e, m\}$, $S_2 = \{a, c, b, c, f, g\}$, $S_3 = \{a, b, c, h, i, m, n\}$ and $S_4 = \{a, b, c, b, c, j, k, m\}$ of lengths 8, 6, 7 and 8, respectively. The matrix M has 8 rows (indexed at the left hand side 1–8). Multiple sequence alignment of the matrix M is the problem of finding a matrix such as M' shown in Fig. 2b in which:

1. Gaps ('—') are inserted among the symbols of the sequences in M.
2. All sequences in M' have the same length (9).
3. Each sequence in M' reverts to its original one in M by removing all the gaps inserted in this sequence.
4. There is no row in M'; which only has the gap character.
5. Number of matched characters (symbols) in M' is maximized.
6. Number of gaps inserted to the matrix is minimized.

After the alignment has been completed, missed values, repetitive patterns and disjunctive patterns could be identified in the aligned matrix. For example, the aligned matrix M' in Fig. 2b shows that the symbol '*a*' is mandatory/required in all the four sequences, the pattern of two symbols { }*b*', }*c*' } is repetitive and has a missing symbol '*b*,' the four patterns { }*d*', }*e*' }, { }*f*', }*g*' }, { }*h*', }*i*' } and { }*j*', }*k*' } are disjunctive, and finally the two symbols '*m*' and '*n*' are missed.

5 The proposed algorithm: NestMSA

Given a peer matrix M that is constructed from a set of sequences $T = \{S_1, S_2, \dots, S_c\}$ as defined above, NestMSA adapts the behavior of PSO to align M and get an aligned matrix such as M' shown in Fig. 2b. Different from all other PSO-based multiple sequence alignment works, NestMSA is iteratively

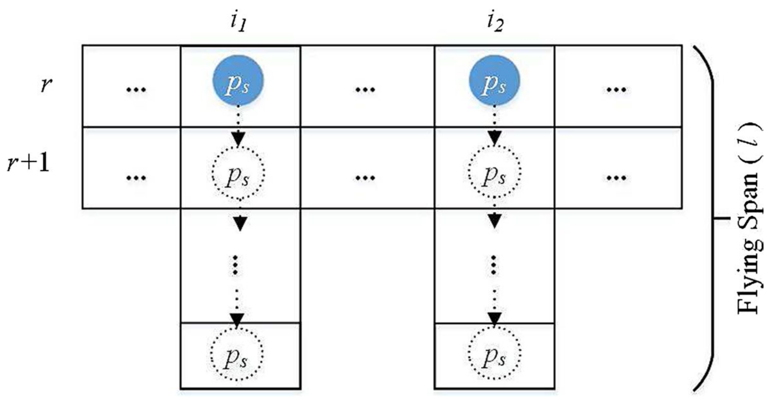


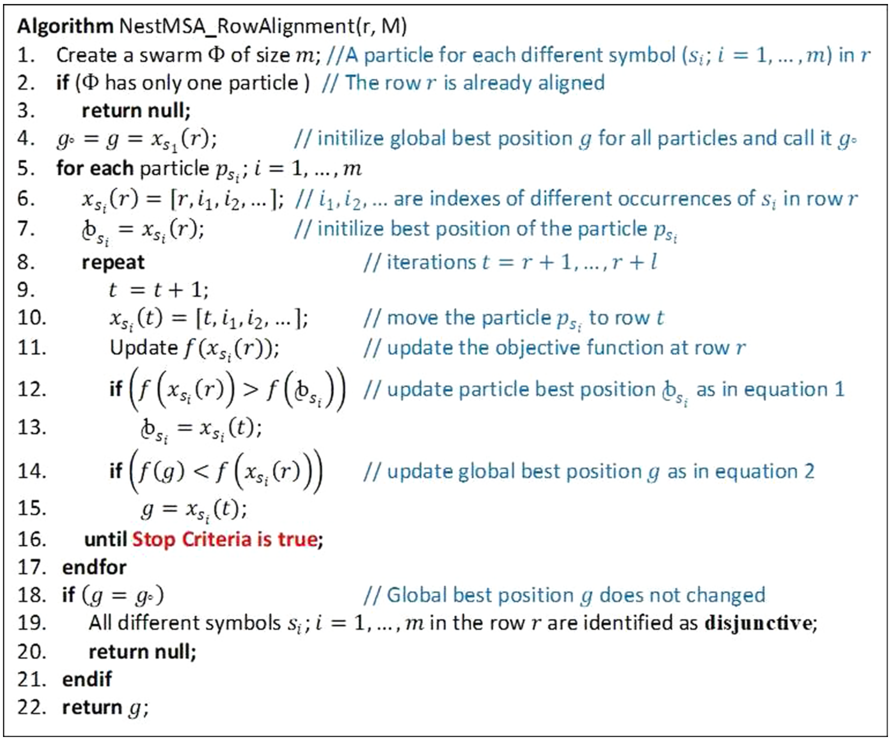
Fig. 1 Particles p_s (for all symbols in a row r) nested at the row to align it

$$\begin{array}{c}1\\2\\3\\4\\5\\6\\7\\8\end{array}\begin{bmatrix} a & a & a & a \\ b & c & b & b \\ c & b & c & c \\ b & c & h & b \\ c & f & i & c \\ d & g & m & j \\ e & \square & n & k \\ m & \square & \square & m \end{bmatrix} \qquad \begin{bmatrix} a & a & a & a \\ b & - & b & b \\ c & c & c & c \\ b & b & - & b \\ c & c & - & c \\ d & f & h & j \\ e & g & i & k \\ m & - & m & m \\ - & - & n & - \end{bmatrix}$$

(a) A peer matrix M **(b)** An aligned matrix M'

Fig. 2 A multiple sequence alignment example

applied to align each row in the matrix. After all rows are aligned, the alignment problem is then solved and the aligned matrix such as M' is obtained. As shown in Fig. 1, for each row r in the matrix M, particles in our algorithm will be initially distributed (nested) over the row r. Then, these particles start flying down to decide which one shall be moved (replaced by gaps). The particle with the global best (if there) shall be moved down (as the particles are distributed over the whole row symbols, so the global best solution is guaranteed). Number of swarm particles m in the row r is less than or equal to number of columns (sequences) c (i.e., $m \leq c$) as each particle p_{s_i} will work with one symbol $s_i (i = 1, \ldots, m)$ in the row, and at the same time, the row may have more than one occurrence of this symbol. That is, all occurrences of the symbol s_i in the row r are carried by a particle p_{s_i}. So, position $x_{s_i}(r)$ of the particle p_{s_i} is defined by using the row r that contains the symbol s_i as well as locations of the symbol s_i in the different columns (indexes of the columns that contain s_i) in the row r. That is:



```
Algorithm NestMSA_RowAlignment(r, M)
1.  Create a swarm $\Phi$ of size $m$; //A particle for each different symbol ($s_i$; $i = 1, \dots, m$) in $r$
2.  if ($\Phi$ has only one particle ) // The row $r$ is already aligned
3.      return null;
4.  $g_\circ = g = x_{s_1}(r)$;          // initilize global best position $g$ for all particles and call it $g_\circ$
5.  for each particle $p_{s_i}$; $i = 1, \dots, m$
6.      $x_{s_i}(r) = [r, i_1, i_2, \dots]$; // $i_1, i_2, \dots$ are indexes of different occurrences of $s_i$ in row $r$
7.      $ƀ_{s_i} = x_{s_i}(r)$;          // initilize best position of the particle $p_{s_i}$
8.      repeat                           // iterations $t = r + 1, \dots, r + l$
9.          $t = t + 1$;
10.         $x_{s_i}(t) = [t, i_1, i_2, \dots]$;     // move the particle $p_{s_i}$ to row $t$
11.         Update $f(x_{s_i}(r))$;                // update the objective function at row $r$
12.         if $\left(f\left(x_{s_i}(r)\right) > f\left(ƀ_{s_i}\right)\right)$ // update particle best position $ƀ_{s_i}$ as in equation 1
13.             $ƀ_{s_i} = x_{s_i}(t)$;
14.         if $\left(f(g) < f\left(x_{s_i}(r)\right)\right)$   // update global best position $g$ as in equation 2
15.             $g = x_{s_i}(t)$;
16.     until Stop Criteria is true;
17. endfor
18. if ($g = g_\circ$)                         // Global best position $g$ does not changed
19.     All different symbols $s_i$; $i = 1, \dots, m$ in the row $r$ are identified as disjunctive;
20.     return null;
21. endif
22. return $g$;
```

Fig. 3 The proposed algorithm (NestMSA_RowAlignment) to align a row r

$$x_{s_i}(r) = [r, i_1, i_2, \dots] \tag{5}$$

where i_1 is the index of the first occurrence of s_i in the row r, i_2 is the index of the second occurrence of s_i in r, and so on. After the m swarm particles $\boldsymbol{\Phi}$ have been nested at row r, they iteratively fly down (row by row at each iteration), keep track and share the global best position which could be used to decide which particle (symbol) should be moved to align the row. Again, this process is repeated until either one particle (symbol) only remains at the row or none of the particles can fly. If one particle only remains at the row, it is considered as an aligned row. If all of the particles cannot fly, we consider the symbols associated with these particles as disjunctive. A symbol will move down to align the row r based on the global best position g of the m particles in the swarm at the row r. Figure 3 shows our proposed algorithm (NestMSA_RowAlignment(r, M)) which is iteratively called to align the row r in the matrix M. It is important to note that, in this section, we use the term move to make an actual modification in the matrix to be aligned, while the term fly shall be used to decide which particle to move (without any actual modification in the matrix). The algorithm is working as follows.

The algorithm starts by initializing the swarm population and creates a set of m particles (line 1) as discussed above. If the swarm has only one particle (i.e., there are different occurrences of only one symbol), the row is considered as aligned and the algorithm is terminated (lines 2–3; no particle will be moved). Otherwise, the algorithm does the following to decide which particle to move. For each particle p_{s_i} $(i = 1, \dots, m)$ in the swarm (lines 5–17), the particle position $x_{s_i}(r)$ is identified as in Eq. 5 and the particle's best position is initialized by the value $x_{s_i}(r)$ (lines 5–7). This means that the initial position of the particle will be at row r (i.e., $t_\circ = r$). Each particle will iteratively fly one step down until stop criteria become true. The position of the particle at each iteration t $(t = r + 1, \dots, r + l)$ is defined as:

$$x_{s_i}(t) = [t, i_1, i_2, \dots];\ \text{where } t = r + 1, \dots, r + l \tag{6}$$

The number of iterations l is called the flying span. This means that the particle p_{s_i} will fly carrying the symbol s_i down in the same column (i.e., only the row will be changed when the particle fly down, while indexes of the columns that contain the symbol are not changed). The flying span l is calculated using a combination of one or more of the following stop criteria:

(i) The flying span (the number of iterations) l is a fixed number. In our experiment, we calculate this fixed value by subtracting the longest sequence (that does not contain the particle symbol s_i) length and the row value r, as the particle does not fly alone. This means the particle p_{s_i} will fly down until no other symbols are shown in the row.

(ii) For each fly of the particle p_{s_i}, the objective function at the new position $(t = r + 1, \dots, r + l)$ does not increase for a fixed value of l. Experimentally, we set the value of l as 5.

(iii) Particle p_{s_i} flies to a new position $x_{s_i}(t) = [t, i_1, i_2, \dots]; t = r + 1, \dots, r + l$, and stops when the symbol s_i appears in the row t at some different indexes j (in other words, there is a column of index $j \notin \{i_1, i_2, \dots\}$ in the matrix at row t which has the symbol s_i). This means the particle p_{s_i} at indexes $i_1, i_2, \dots$ is flying until it finds the symbol s_i down in a row at an index $j \neq i_k; k = 1, 2, \dots$, otherwise the stop criteria ii and i are applied, in order.

At each iteration t (ranged from r to $r + l$), the particle position is modified by increasing the row number by 1 (lines 9–10). The objective function at the next row (will be discussed later) is updated (line 11) after the particle symbol is flying down to row t (i.e., after the matrix has been virtually modified). Then, the value of the new objective function will be used to calculate both the particle best position $\mathfrak{I}_i$ (lines 12–13) as in Eq. 1 and the global best position for all particles as in Eq. 2 (lines 14–15). The global best position was initialized before the particles start flying. This initial global best value is called $g_\circ$ in the algorithm (line 4). After the global best position $g = [t', i_1, i_2, \dots]$ is tracked/shared by all particles, it is returned by the algorithm (line 22). Hence, all occurrences (indexed by $i_1, i_2, \dots$) of the symbol with this global best position will be moved down a distance equal to $(t' - r)$ in the matrix M. At the same time, each gap added to the matrix is replaced by—. If the global best position does not change (i.e., $g = g_\circ$), the algorithm is terminated after recognizing all the different symbols $s_i; i = 1, \dots, m$ in the row r as disjunctive/mutative (lines 18–21; no particle will be moved).

As discussed above, the proposed algorithm adapts the particle swarm optimization as follows. First, it simply considers the particles initial positions as the row to be aligned. Second, the velocity of the particle at iteration $t+1$ will improve the particle position at iteration $t+1$ by just changing the row value t in the particle position $x_s(t) = [t, i_1, i_2, \ldots]$ at iteration t. This means that the flying directions of the particles depend only on the inertia component and ignored both the cognitive components and the social component.

To analyze the algorithm that aligns a row r of m different symbols, we consider the number of comparisons between two symbols as the cost measure to determine the running time. Updating the objective function at line 11 in Fig. 3 requires $(c-1) \times (k-r)$ comparisons (see Eq. 8 in the next subsection), where c and k are the number of columns (sequences) and the longest sequence length, respectively. The inner loop (lines 8–16 in the figure) stops after l iterations (flying span) which is calculated based on the three criteria mentioned above. So, the inner loop requires $l \times (c-1) \times (k-r)$ comparisons. This will be repeated (the outer loop—lines 5–17) for each different symbol (particle) in r (i.e., the outer loop has m iterations). Experimentally, as mentioned above, the flying span l based on the three criteria applied in the algorithm is less than 5. Therefore, the running time complexity of the algorithm is $O(m \times c \times k)$.

5.1 Particle objective function

Before we go further in this subsection to discuss how to calculate the objective function of a particle p_s at position $x_s(t) = [t, i_1, i_2, \ldots]$ in a matrix M, we introduce some important definitions.

Definition 3 (*Aligned row*) Given a matrix M as discussed in Sect. 4, a row r is called an aligned row if it only contains different occurrences of the same symbol s.

Definition 4 (*Full row*) An aligned row r is called full if no gaps (—) are added in the row r. That is, the number of occurrences of the symbol in the row is equal to the number of columns in the matrix.

Definition 5 (*Row weight*) Weight of the row r is calculated as:

$$\omega(r) = \begin{cases} \omega_1 \times \frac{x}{c}; & \text{if } r \text{ is not aligned} \\ \omega_2 \times \frac{\hbar_s}{c}; & \text{if } r \text{ is aligned} \\ \omega_3; & \text{if } r \text{ is Full} \end{cases} \tag{7}$$

where ω_1, ω_2 and ω_3 are the penalties of the row if it is not aligned, aligned and full, respectively. Experimentally, as we shall mention later, we chose the values for these penalties as: $\omega_1 = 0.25$ (for the non-aligned row), $\omega_2 = 0.5$ (for the aligned row) and $\omega_3 = 1$ (for a full aligned row), although other values could be more suitable for other experiments. For example, when evaluating the proposed MSA algorithm in the field of Web data extraction, we used a higher value of ω_3 ($\omega_3 = 2$) if the symbol in a full aligned row corresponds to a leaf node in the DOM tree which is a part of the Web page template (not classified as data to be extracted). The value n_s is the number of occurrences of the symbol s in the aligned row r, and c is the total number of columns in the matrix M ($c > 1$). The value x is calculated for the non-aligned row r as follows. The value of x is equal to zero if every symbol in the row r occurred at most once, otherwise x is equal to the max number of occurrences (matches) of some symbol in r. The proposed algorithm is a single objective optimization problem. The objective function f of the particle p_s at position $x_s(r)$ in the matrix M is calculated based on the accumulated weights of all rows from r to the last row ($\sum_{j=r}^{k} \omega(j)$). Also, as discussed in Sect. 4, the objective function has a converse relationship with the number of gaps $Gaps(r)$ added to the matrix from row r to the last row, and a positive relationship with the number of aligned rows $A(r)$ in M from the row r to the last row (k). Finally, we define a row context value $C(r)$ as the maximum number of matched characters (symbols) in the current row r ($C(r) \geq 1$). The objective function f should be increased when $C(r)$ is increased. Therefore, the objective function is formulated as follows.

$$f(x_s(r)) = \frac{A(r) \times C(r)}{1 + Gaps(r)} \times \sum_{j=r}^{k} \omega(j) \tag{8}$$

5.2 An illustrative example

To clarify our proposed algorithm in Fig. 3, it will be applied to solve the alignment problem on the fictional matrix M shown in Fig. 2a. To align the first row ($r = 1$), the algorithm NestMSA_RowAlignment will create a swarm $\boldsymbol{\Phi}$ that includes only one particle p_a as the row has only one symbol a. So, it returns null because the row is already aligned (lines 2–3 in Fig. 3). To align the second row ($r = 2$), a swarm $\boldsymbol{\Phi}$ that contains the two particles p_b and p_c is created as the row has two symbols (b and c). To decide which symbol to move to align this row, each of the two particles will fly down in the matrix iteratively and the global best position will be tracked and calculated. Based on this calculated global best position, the algorithm will decide which particle (symbol) to move in order to align the row. Sometimes, the algorithm is called more than one time to align the row. The initial global best position at the second row is $g_\circ = x_b(2) = [2, 1, 3, 4]$ where the symbol 'b' has occurred at indexes 1, 3 and 4 at the row. Objective function (Eq. 8) at the initial position $g_\circ$ is calculated as:

$f(g_\circ) = \frac{1\times3}{1+0} \times 0.875 = 2.625$,

where $\sum_{j=2}^{8} \omega(j) = 0.25 \times \frac{3}{4} + 0.25 \times \frac{3}{4} + 0.25 \times \frac{2}{4} + 0.25 \times \frac{2}{4} + 0 + 0 + 0.5 \times \frac{2}{4} = 0.875$, $A(2) = 1$ (as there is only one aligned row, the eighth row, in the range from row 2 to row 8) and $Gaps(2) = 0$ (as no gaps added to the rows from 2 to 8 till now). Also, the row context value is $C(r) = 3$, because the symbol 'b' has a maximal occurrence value (3 occurrences in the second row). Now, either of the two particles p_b or p_c is

$$\begin{array}{r|cccc|}
1 & a & a & a & a \\
\Rightarrow 2 & - & c & - & - \\
3 & b & b & b & b \\
4 & c & c & c & c \\
5 & b & f & h & b \\
6 & c & g & i & c \\
7 & d & \square & m & j \\
8 & e & \square & n & k \\
9 & m & \square & \square & m
\end{array}$$

(a) M after particle p_b at 2nd row flies one row down $f(x_b(2)) = 2.625$

$$\begin{array}{r|cccc|}
 & a & a & a & a \\
\Rightarrow & b & - & b & b \\
 & c & c & c & c \\
 & b & b & h & b \\
 & c & c & i & c \\
 & d & f & m & j \\
 & e & g & n & k \\
 & m & \square & \square & m \\
 & \square & \square & \square & \square
\end{array}$$

(b) *M after particle p_c at 2nd row flies one row down* $f(x_c(2)) = 6.0$

$$\begin{array}{r|cccc|}
1 & a & a & a & a \\
2 & b & - & b & b \\
3 & c & c & c & c \\
\Rightarrow 4 & b & b & - & b \\
5 & c & c & - & c \\
6 & d & f & h & j \\
7 & e & g & i & k \\
8 & m & \square & m & m \\
9 & \square & \square & n & \square
\end{array}$$

(c) M after particle p_h at 4th row flies 2 rows down $f(x_h(4)) = 5.0$

$$\begin{bmatrix}
a & a & a & a \\
b & - & b & b \\
c & c & c & c \\
b & b & - & b \\
c & c & - & c \\
d & f & h & j \\
e & g & i & k \\
m & - & m & m \\
- & - & n & -
\end{bmatrix}$$

(d) *M after alignment has been finished and Gaps added to have sequences of similar lengths.*

Fig. 4 Snapshots of the matrix M shown in Fig. 1a during the alignment process

flying down if a global best position g is found such that $f(g) > f(g_\circ) = 2.625$, otherwise no flying is done.

To decide which symbol to move in the second row, the first particle will start flying and it stops after 1 iteration at $t = 3$ (applying the third stop criterion iii discussed above). Figure 4a shows the matrix M after flying the particle p_b to the third row. At that time, the particle new position is $x_b(3) = [3, 1, 2, 3, 4]$, the objective function $f(x_b(2))$ after the particle flying is $f(x_b(2)) = \frac{4\times1}{1+3} \times 2.625 = 2.625$, and so the particle's best and the global best positions will not be changed as the objective function when the particle p_b is flying down is the same as the objective function at its initial position (i.e., $f(g) = f(g_\circ) = 2.625$. Note that the algorithm will use a copy of the matrix M to be modified when particles are flying and calculate the global best position without changing the original matrix.

For the second particle p_c in the swarm, it also flies and stops after 1 iteration. Figure 4b shows the matrix M after the particle p_c flies to the third row. At that time $x_c(3) = [3, 1, 2, 3, 4]$, $f(x_c(2)) = \frac{3\times3}{1+1} \times 2.0 = 6.0$, and so the particle best and the global best positions will be: $\mathfrak{I}_c = x_c(3)$ and $g = x_c(3)$, respectively. Since the objective function at the global best position is greater than the one at the initial position (i.e., $f(g) = 6.0 > f(g_\circ) = 2.625$), the algorithm will return the last modified global best $g = x_c(3) = [3, 1, 2, 3, 4]$. Therefore, after all particles have been communicated, the second row will be aligned by moving the symbol ('c') at index 2 a distance $3 - 2 = 1$. This means that the second row will be aligned by moving 'c' to the third row and adding a gap at that place to get a matrix M shown in Fig. 4b.

Aligning the second row shown in Fig. 4 is a good example which demonstrates the strength of the proposed objective function in the algorithm. Comparing the two matrixes in Fig. 4a, b, each of the two matrixes could be a solution to our alignment problem. The objective function at the second row will take the responsibility to choose one of these two matrixes (the one that maximizes the objective function value). That is why, this function may need to be adapted when applying to other fields. It is clear from the calculated objective function values at the second row that the more aligned rows and the minimum added gaps give the larger objective value.

Similar to the first row, the swarm for the third row has only one particle p_c, so it returns null as the row is aligned (see Fig. 4b). For the fourth row, the swarm includes the two particles p_b and p_h. The initial global best position is $g_\circ = x_b(4) = [4, 1, 2, 4]$, where the symbol '$b$' at the fourth row has indexes 1, 2 and 4, and then $f(g_\circ) = (1 \times 3)/(1 + 0) \times 0.625 = 1.875$. Similar to aligning the second row, using the first stop criterion (i), the particle p_b will fly 3 times (r =5,6 and 7). After the particle p_b stops flying, the global best position will be $g = x_b(7) = [7, 1, 2, 4]$. After the particle p_h stops flying, the global best is updated to be $g = x_h(6) = [6, 3]$ at which $f(x_h(4)) = (4 \times 3)/(1 + 2) \times 1.25 = 5.0$. So, the algorithm at the fourth row of the matrix M shown in Fig. 4b will return $g = x_h(6) = [6, 3]$. This means that the row will be aligned by moving the symbol 'h' at the fourth row a distance $6 - 4 = 2$ and adding gaps in that places. The matrix obtained after this step is shown in Fig. 4c.

In the sixth row of the matrix in Fig. 4c, the swarm creates the four particles: p_d, p_f, p_h and p_j. The initial global best position is $g\circ = x_d(6) = [6, 1]$ where the symbol 'd' at the sixth row has an occurrence at index 1, and then $f(g_\circ) = 1.0$. Each of the four particles will fly down, but the global best and the local best will not be modified as none of them fly to a position at which the objective function is greater than $f(g_\circ) = 1.0$. This means that the global best for the four particles remains $g_\circ$. Therefore, the algorithm terminates after identifying the four symbols 'd,' 'f,' 'h' and 'j' as disjunctive/mutative. Similarly, the algorithm identifies the four symbols 'e,' 'g,' 'i' and 'k' as disjunctive in the seventh row. Finally, the last two rows are aligned. Gaps are added in the row when it is identified by the algorithm as an aligned row and it was not full. The result will be the matrix shown in Fig. 4d which is the same as the one in Fig. 2b (M').

6 Experiments

In this paper, we focused on measuring the performance of the proposed algorithm and comparing it with other algorithms in the two domains: Web data extraction and removing DUST, although it could be adapted to be used in other domains. Sequences in these domains are usually of small sizes. Each character in the sequence corresponds to either a datum to be extracted (DOM subtree) from Web pages or an URL token as mentioned before. Three experiments are conducted in this paper, two of them are used to evaluate the algorithm in the field of Web data extraction, while the third one is used to evaluate the algorithm in the domain of removing DUST. The first experiment is conducted to measure the performance of the proposed algorithm on a dataset of peer matrixes that are generated by a Web data extraction system. The second experiment tries to compare the proposed algorithm with the two alignment algorithms in TEX [19] and DCA [20]. Finally, the third experiment is conducted to measure the performance of the algorithm on a set of dup-clusters (each dup-cluster is a set of different URLs with similar contents) gathered from the common crawl dataset. All the experiments are conducted on a standard Intel Core i7-4600U 2.7 GHz CPU with 8G RAM running window 10.

Recall and precision are used to measure the performance of the algorithm in the first two experiments. We use the same formulas applied in both DCA and TEX in order to achieve the comparison. Given an aligned matrix M'_{alg} extracted by an algorithm called '*alg*' and the aligned matrix M'_{manually} extracted manually by experts, recall (R_r) and precision (P_r) for each row r in the aligned matrix M'_{alg} are calculated as follows:

$$R_r = \frac{\#\text{correct aligned symbols in } M'_{\text{alg}} \text{ at row } r}{\#\text{ symbols in } M'_{\text{manually}} \text{ at row } r}$$

$$P_r = \frac{\#\text{correct aligned symbols in } M'_{\text{alg}} \text{ at row } r}{\#\text{ symbols in } M'_{\text{alg}} \text{ at row } r}$$

Now, recall (R) and precision (P) could be calculated for each matrix by dividing the number of correctly extracted rows (CR) by the total number of rows in M'_{manually} (GA) and the total number of rows in M'_{alg} (ER), respectively. Similar to DCA, we consider a row as correctly extracted if $R_r \geq 0.85$. Therefore, recall (R), precision (P) and F-measure (F) are calculated as follows:

$$R = \frac{CR}{GA}, \quad P = \frac{CR}{ER}, \quad F = \frac{2 \times R \times P}{R + P} \tag{9}$$

For the first experiment, a dataset is generated for the purpose of this work by the help of a Web data extraction system [3, 26] and a simple GUI. The tools developed for the Web data extraction system FiVaTech are used to automatically collect all the matrixes to be aligned. The research purpose-designed GUI is used to simplify the manual labeling and alignment process of the collected matrixes. (C++ is the core language for this GUI as FiVaTech is written using it.) A total of 1200 peer matrixes

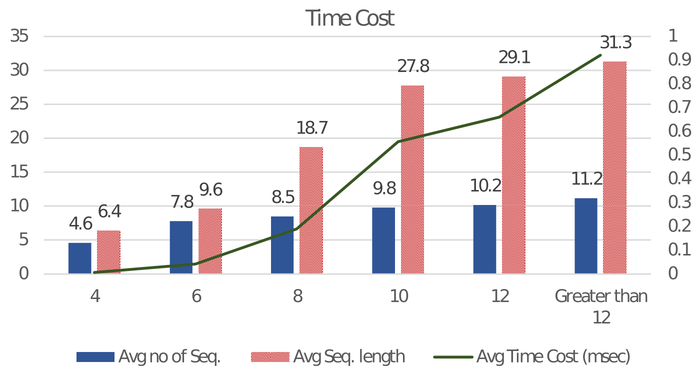


Fig. 5 Time cost for the algorithm NestMSA

taken from 51 Web sites (Testbed for Information Extraction from Deep Web (TBDW) Version 1.02 dataset[1]) are collected, and their corresponding alignments are created to measure the performance of the proposed algorithm. We ignore to collect peer matrixes formed from subtrees near the roots of the DOM trees as many of them already have aligned rows (full aligned in many cases). We add a constraint on the characters (data) in the peer matrixes to be taken from DOM trees at least 4 levels from the root (<*body*> tag). Particularly, we take 200 peer matrixes from each level (4, 6, 8, 10 and greater than 10) in the DOM trees (a total of 1200 matrixes). Figure 5 shows the average time cost for each block of 200 matrixes (in milliseconds). Also, both the average number of sequences and the average sequence lengths for each block are shown in the figure. As shown in the figure, the time cost to align peer matrixes near the root of the DOM trees is very small and it is increased gradually when the dimensions of the matrixes increased. The average time cost for the whole 1200 matrixes is approximately equal to 0.42 ms.

To guarantee that the results are not biased to the used dataset, we used the following heuristic to find the most appropriate penalties for the non-aligned, aligned and full aligned rows. We fix the value of ω_2 to be 0.5, start ω_1 and ω_3 with the values 0.1 and 0.6, respectively, and finally measure the precision and recall. After that, iteratively we add 0.05 to each of the two values of ω_1 and ω_3, and again measure the precision and recall at each iteration. Finally, we chose the penalties values with the maximum F-measure. If more than one iteration has a maximum F-measure, we chose the highest ω_1 and ω_3 values.

Table 1 shows the results of applying our proposed algorithm on the dataset in three different cases when each of the three stopping criteria discussed in Sect. 5 is used to calculate the particles' flying span (columns 2, 3 and 4), respectively. For each case, the total number of gaps, the total number of matched symbols, the total number of disjunctive symbols, the total number of full aligned rows and the average alignment score are calculated and compared with the manually generated aligned

[1] http://daisen.cc.kyushu-u.ac.jp/TBDW/.

Table 1 Results of applying the algorithm on the dataset using 3 different stop criteria

	Criterion i	Criterion ii	Criterion iii	Manual alignment
# Gaps	4750	4738	4523	4496
# Matched symbols	171765	171778	172047	172900
# Disjunctive	432	421	388	382
# Full Aligned Rows	6138	6119	6128	6115
Avg score of the aligned matrixes	14.6	14.8	15.2	14.9

Table 2 The performance of the proposed algorithm

	Criterion i	Criterion ii	Criterion iii
R	95.3	95.1	96.7
P	93.9	93.7	95.8
F	94.6	94.4	96.2

matrixes. The alignment score is calculated using Eq. 8 for each matrix (resulted by the proposed algorithm), and then the average of these matrixes scores is calculated. We set 1 to the value $C(r)$ in Eq. 8. The total number of gaps is calculated after ignoring gaps added at the end of the sequences to have the same length. Also, when calculating the number of matched symbols, we only consider the rows that have more than one occurrence of a symbol. As shown in Table 1, applying the last criterion gives better results. The performance of the algorithm in the three different stopping cases is measured by using recall, precision and F-measure (defined in Eq. 9). Table 2 shows that there is no significant difference when applying either of the first two criteria (i and ii), while the last criterion (iii) gives better performance.

For the second experiment, we use the 22 Web sites in the DCA dataset[2] that are used for the comparison between TEX and DCA. The average number of pages in each site is 30 pages, while the average number of sequence lengths is 292. To compare our proposed algorithm with the two algorithms DCA and TEX, we used recall, precision and F-measure that are defined in Eq. 9. Table 3 shows that our proposed algorithm outperforms both DCA and TEX. The performance of the proposed algorithm is: $R = 0.95$, $P = 0.96$, $F = 0.96$. The average number of the manually aligned rows (called golden answer in DCA) is 47 rows per website, while the average number of aligned rows extracted by the proposed algorithm is 47.27 which is much closer. TEX suffered from false-positive data attributes. It produced the highest number of aligned rows, resulting in low precision.

Finally, the last experiment is conducted on a collection Web pages URLs crawled from the common crawl dataset,[3] which is an open repository of Web crawl data. We developed a simple tool to choose only Web sites supporting Google's

[2] https://sites.google.com/site/nculab/project/WDEMS/dca.

[3] http://commoncrawl.org/the-data/get-started/.

Table 3 Performance comparison between our algorithm and both DCA and TEX

Site	GA	DCA				TEX				Our algorithm			
		ER	R	P	F	ER	R	P	F	ER	R	P	F
1	44	43	0.98	0.97	0.97	52	0.59	0.63	0.61	44	1.00	1.00	1.00
2	21	21	1.00	1.00	1.00	37	0.76	0.64	0.69	21	1.00	1.00	1.00
3	39	37	0.93	0.97	0.95	44	0.33	0.34	0.33	40	0.90	0.88	0.89
4	117	113	0.96	0.98	0.97	156	0.54	0.55	0.54	115	0.94	0.96	0.95
5	101	100	0.87	0.93	0.90	119	0.36	0.35	0.35	100	0.94	0.95	0.95
6	51	43	0.84	0.98	0.90	93	0.84	0.52	0.64	48	0.86	0.92	0.89
7	92	90	0.93	0.96	0.94	101	0.65	0.67	0.66	90	0.97	0.99	0.98
8	13	13	1.00	1.00	1.00	13	0.85	0.61	0.71	13	1.00	1.00	1.00
9	15	16	0.91	0.86	0.88	11	0.87	0.77	0.82	15	1.00	1.00	1.00
10	21	19	0.75	0.83	0.79	24	0.67	0.70	0.68	20	0.86	0.90	0.88
11	31	30	0.96	0.97	0.96	64	0.58	0.49	0.53	29	0.87	0.93	0.90
12	106	106	0.93	0.95	0.94	54	0.52	0.65	0.58	107	0.99	0.98	0.99
13	14	14	1.00	1.00	1.00	52	0.43	0.12	0.19	14	1.00	1.00	1.00
14	48	48	0.97	0.97	0.97	55	0.83	0.70	0.76	48	1.00	1.00	1.00
15	38	37	0.93	0.97	0.95	35	0.74	0.78	0.76	38	0.95	0.95	0.95
16	13	13	1.00	1.00	1.00	22	0.92	0.60	0.73	13	1.00	1.00	1.00
17	17	17	1.00	1.00	1.00	17	0.94	1.00	0.97	17	1.00	1.00	1.00
18	70	67	0.91	0.96	0.93	66	0.73	0.70	0.71	71	0.97	0.96	0.96
19	36	35	0.96	0.99	0.97	21	0.92	0.89	0.90	35	0.92	0.94	0.93
20	30	30	0.95	0.97	0.96	53	0.71	0.44	0.54	28	0.93	1.00	0.97
21	60	60	0.98	0.98	0.98	83	0.70	0.54	0.61	57	0.83	0.88	0.85
22	78	78	0.96	0.96	0.96	104	0.64	0.50	0.56	77	0.95	0.96	0.95
Avg	47.95	46.82	0.94	0.96	0.95	58.00	0.69	0.60	0.63	47.27	0.95	0.96	0.96

Table 4 Performance of the proposed algorithm in removing dust domain

Site	*R*	*P*	*F*	SPS	Site	*R*	*P*	*F*	SPS
1	0.92	0.90	0.91	0.88	6	0.95	0.94	0.94	0.93
2	0.96	0.92	0.94	0.92	7	1.00	0.98	0.99	0.95
3	0.94	0.94	0.94	0.95	8	0.93	0.90	0.91	0.87
4	0.90	0.89	0.89	0.90	9	0.94	0.93	0.93	0.93
5	0.96	0.91	0.93	0.92	10	0.95	0.96	0.95	0.91
Average						0.95	0.93	0.94	0.92

suggestion [27], in which the canonical URLs are explicitly specified. The data are collected from 10 Web sites. (The 10 sites are extracted using the three categories: Sports, Travel and Business.) The average number of dup-clusters in a Web site is 275, while the total number of dup-clusters (peer matrixes) in the ten sites is 3170. The minimum number of URLs in a dup-cluster is 2, and the maximum number is 49. Our aim in this experiment is evaluating the performance of the proposed MSA algorithm in the DUST domain; we are not focusing on finding or evaluating the normalization rules that transform a group of URLs having the same formation into a canonical formation. We used the same measures used in the two experiments (*R*, *P* and *F*) discussed before as well as the common score metric Sum of Pairs (SPS). Table 4 shows the results of the 10 Web sites. As shown in the table, the performance of the algorithm is high (recall, precision and F averages are 0.95, 0.93 and 0.94, respectively). The SPS score values (the average is 0.92) are calculated based on the distance scoring among the sequences.

7 Conclusion

This paper suggested a new multiple sequence alignment algorithm (NestMSA) that is inspired by the particle swarm optimization behavior. Different from all other researches that adapt PSO to solve the multiple sequence alignment problem, particles in our proposed algorithm are nested and communicated among the sequences (at one row in the matrix) to guarantee obtaining the global maximum. A new objective function is also suggested which considers all the characteristics of the multiple sequence alignment problem. To improve the efficiency of the algorithm, a combination of three different stopping criteria are applied to reduce the particles' flying span. The proposed algorithm considered the variations occurred in the multiple sequence alignment problem such as missing patterns, repetitive patterns and disjunctive patterns. We addressed the problem of MSA in the two domains Web data extraction and removing DUST as very few works are done although those are very important domains. But, we will consider the problem of DNA sequences alignment in which the sequences are very long with multiple nested repetitive regions in our future work.

References

1. Mount DW (2004) Bioinformatics: sequence and genome analysis, 2nd edn. Cold Spring, Harbor, NY
2. Zhai Y, Liu B (2005) Web data extraction based on partial tree alignment. In: Proceeding of the International Conference of World Wide Web (WWW-14), pp 76–85
3. Kayed M, Chang C-H (2010) GiVaTech: page-level web data extraction from template pages. IEEE Trans Knowl Data Eng 22(2):249–263
4. Rodrigues K, Cristo M, De Moura ES, Da Silva A (2015) Removing DUST using multiple alignment of sequences. IEEE Trans Knowl Data Eng 27(8):2261–2274
5. Bar-Yossef Z, Keidar I (2009) Do not crawl in the DUST: different URLs with similar text. In: ACM Transactions on the Web (TWEB), vol 3. New York, USA, pp 1–31
6. Kayed M (2012) Peer matrix alignment: a new algorithm. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining (ICDM), pp 268–279
7. Notredame C (2002) Recent progress in multiple sequence alignment: a survey. Pharmacogenomics 3(1):131–144
8. Rubio-Largo A, Vanneschi L, Castelli M, Vega-Rodrguez MA (2018) A characteristic-based framework for multiple sequence aligners. IEEE Trans Cybern 48(1):41–51

9. Quoc-Nam T, Wallinga M (2017) UPS: a new approach for multiple sequence alignment using morphing techniques. In: IEEE International Conference on Bioinformatics and Biomedicine (BIBM), pp 425–430
10. Thompson JD, Gibson TJ, Plewniak F, Jeanmougin F, Higgins DG (1997) The CLUSTAL X windows interface: flexible strategies for multiple sequence alignment aided by quality analysis tools. Nucleic Acids Res 25(24):4876–82
11. Notredame C, Higgins DG, Heringa J (2000) T-COFFEE: a novel method for fast and accurate multiple sequence alignment. J Mol Biol 302(1):205–217
12. Rodriguez PF, Nino LF, Alonso OM (2007) Multiple sequence alignment using swarm intelligence. Int J Comput Intell Res 3(2):123–130
13. Xu F, Chen Y (2009) A method for multiple sequence alignment based on particle swarm optimization. In: 5th International Conference on Intelligent Computing, ICIC 2009, Ulsan, South Korea, pp 16–19
14. Zhan Q, Wang N, Jin S, Tan R, Jiang Q, Wang Y (2019) ProbPFP: a multiple sequence alignment algorithm combining hidden Markov model optimized by particle swarm optimization with partition function. BMC Bioinf 20:573
15. Chang C-H, Kuo S-C (2004) OLERA: a semi-supervised approach for Web data extraction with visual support. IEEE Intell Syst 19(6):56–64
16. Crescenzi V, Mecca G, Merialdo P (2001) RoadRunner: towards automatic data extraction from large web sites. In: 27th international conference on very large data bases. Morgan Kaufmann Publishers Inc
17. Simon K, Lausen G (2005) ViPER: augmenting automatic information extraction with visual Perceptions. In: Proceedings of the 2005 ACM CIKM international conference on information and knowledge management, Bremen, Germany, pp. 381–388, October 31–November 5, 2005
18. Yuliana OY, Chang C-H (2016) AFIS: aligning detail-pages for full schema induction. In: Conference on technologies and applications of artificial intelligence (TAAI)
19. Sleiman HA, Corchuelo R (2013) TEX: an efficient and effective unsupervised web information extractor. Knowl Based Syst 39:109–123
20. Yuliana OY, Chang C-H (2018) A novel alignment algorithm for effective web data extraction from singleton-item pages. Appl Intell 48(11):4355–4370
21. Chentoufi A, El Fatmi A, Bekri A, Benhlima S, Sabbane M (2017) Genetic algorithms and dynamic weighted sum method for RNA alignment. In: Intelligent Systems and Computer Vision (ISCV) Conference, pp 1–5
22. Amorim AR, Visotaky JMV, Contessoto ADC, Neves LA., De Souza RCG, Valncio RC, Zafalon GFD (2017) Performance improvement of genetic algorithm for multiple sequence alignment. In: 17th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT), pp 69–72
23. Lalwani S, Kumar R, Gupta N (2013) A review on particle swarm optimization variants and their applications to multiple sequence alignment. J Appl Math Bioinf 3(2):87–124
24. Jagadamba PVSL, Babu MSP, Rao AA, Rao PKS (2011) An improved algorithm for multiple sequence alignment using particle swarm optimization. In: IEEE 2nd International Conference on Software Engineering and Service Science, pp 544–547
25. Kamal A, Mahroos M, Sayed A, Nassar A (2012) Parallel particle swarm optimization for global multiple sequence alignment. Inf Technol J 11(8):998–1006
26. Chang C-H, Chang C-H, Kayed M (2012) Fivatech2: a supervised approach to role differentiation for web data extraction from template pages. In: 26th Annual Conference of the Japanese Society for Artificial Intelligence, Special Session on Web Intelligence & Data Mining, pp 1–9
27. Google webmaster central blog: specify your canonical. https://webmasters.googleblog.com/2009/02/specify-your-canonical.html