

Sequence Alignment Guided By Common Motifs Described By Context Free Grammars

Abdullah N. Arslan
Department of Computer Science
University of Vermont
Burlington, VT 05405, USA
Email: aarslan@cs.uvm.edu

Abstract—We introduce a new problem, *context-free grammars (CFG)-guided pairwise sequence alignment*, whose most immediate application is the alignment of *RNA* sequences that share motifs described by context-free grammars. Such motifs include common *RNA* secondary (sub)structures (such as stem-loops) that are recognizable in sequences. The problem aims to align given sequences by including, from a given set of weighted motifs, any number of motifs in any common order possible in the sequences aligned to maximize the resulting score. The score is a function of included motifs (there is a given score for each motif which is the same for every variation of the same motif), and a set of ordinary pairwise sequence alignments where these motifs are not included (non-motif-matching regions). The main differences of this new problem from existing constrained sequence alignment problems are that the new problem allows using more complex motifs that are described by context free grammars, and considers multiple such motifs that can be found in given sequences in many different orders from which a score-optimal order is sought. We present a dynamic programming solution for this problem. Our definitions and results can be generalized to local and/or multiple sequence alignment problems guided by context free grammars for various scoring schemes.

Keywords: RNA, motif, sequence alignment, constrained sequence alignment, context free grammar, parsing, dynamic programming.

I. INTRODUCTION

Given two strings S_1 , and S_2 , the *pairwise sequence alignment* can be described as a writing scheme such that we use a two-row-matrix in which the first row is used for the symbols of S_1 , and the second row is used for those of S_2 , and each symbol of one string can be aligned to (i.e. it appears on the same column with) a symbol of another string, or the blank symbol '-'. A matrix obtained this way is called an *alignment matrix*. No column can be entirely composed of blank symbols. A given function assigns a weight to each column. The score of an alignment is the total score of the columns in the corresponding alignment matrix. This process can be generalized to alignment of multiple sequences.

Ordinarily, in sequence alignment we would like to find an alignment with maximum score. Constrained versions of sequence alignment problems have been studied in the literature extensively [3], [5], [6], [10], [26], [27], [28]. The motivation for these problems has been that biologically meaningful alignments contain regions that indicate a biological function

or signature, *motif*. This does not always yield increased scores. Comet and Henry [10] experimentally showed that the Smith-Waterman [31] local alignment algorithm does not always align motif-regions together. In addition, biologists would like to incorporate their knowledge about common structures into the alignment process. Forcing the alignments align such common motifs together guides the alignment process. A summary of evolution of motif definitions, and their use in sequence alignment can be found in [1]. A motif can be a string which is required to be included as a subsequence in an alignment sought exactly [3], [5], [6], [10], [26], [27], [28] or within a given tolerance [3]. Motifs can also be patterns that are described by regular expressions [10], [1].

In this paper, we propose a method that aligns sequences guided by a given set of weighted motifs each of which is described by a *context free grammar (CFG)*.

Let $\mathcal{A}(u, v)$ denote the maximum ordinary alignment score for strings u and v that can be computed using the Needleman-Wunsch algorithm [22]. We introduce *context-free grammars (CFG)-guided pairwise sequence alignment (CFG-PSA)* as the following problem: given two sequences S_1 and S_2 , and a set of context-free grammars $\mathcal{C} = \{\mathcal{G}_1, \dots, \mathcal{G}_f\}$ each of which represents a motif whose weight is specified by a given function β , compute

$$\max \sum_{i=1}^{k+1} \mathcal{A}(x_i, w_i) + \sum_{y_i, z_i \in \mathcal{G}, \mathcal{G} \in \mathcal{C}} \beta(\mathcal{G})$$

over all possible $y_1, y_2, \dots, y_k, z_1, z_2, \dots, z_k$ where $S_1 = x_1 y_1 x_2 y_2 \dots x_k y_k x_{k+1}$, $S_2 = w_1 z_1 w_2 z_2 \dots w_k z_k w_{k+1}$, each x_j or w_j , $1 \leq j \leq k+1$, can be an empty string, and for all i , $1 \leq i \leq k$, there exists a $\mathcal{G} \in \mathcal{C}$ such that $y_i \in L(\mathcal{G})$, and $z_i \in L(\mathcal{G})$, where $L(\mathcal{G})$ is the language generated by *CFG* \mathcal{G} . We call each y_i in S_1 , and z_i in S_2 as *motif-matching substring*, and a y_i - z_j -pair together as a *motif-matching region*, or simply a *motif-match*.

Motivations for this definition are as follows: Many interesting *RNA* molecules conserve a secondary structure of base-pairing interactions more than they conserve their sequences. Common base-pairing interactions (such as stem-loops in *RNA* secondary structures) identifiable in *RNA* sequences can be described by context free grammars [13]. We extend the definition of a motif to include such sequence structures. We

note that this definition is powerful enough to describe regular expression motifs, and subsequence motifs (a subsequence is a special regular expression) because of the hierarchy of classes of formal languages [17]. A motif can have variations (e.g. the *K-turn* motif [19]). However, since the union of two *CFG*'s is a *CFG* [17], all these variations can be described by a single *CFG*. We assume that we are given a set of weighted *CFG*'s each of which represents a motif with all its possible variations. A pair of substrings that are generated by the same *CFG* (a motif-match) yields a score that is the assigned weight to that *CFG*. *RNA* sequences may contain varying number of such motifs in different orders (e.g. the *K-turn* motif [19]). In our definition, the purpose of the alignment process is to align sequences in a way to include all or some of the motif-matches in an order to optimize the resulting score.

In Figure 1, for illustrative purposes we imagine two short syntectic *RNA* sequences $S_1 = AACGGAACGGCAAAAACUUUUUAUACCCGUGC$, and $S_2 = AAGGGAACCGACAUAUAUGUAUCGCGGACGC$. In the alignment in Part (c), we distinguish the motif-matching regions by enclosing them by rectangles in dashed lines. Each of these regions has a score determined by a given function β which assigns a score for each motif. The remaining regions in each sequence are aligned pairwise using ordinary sequence alignment. The score of the alignment shown in Part (c) is $\mathcal{A}(AA, AA) + \mathcal{A}(GCA, GA) + \mathcal{A}(UAU, UA) + \mathcal{A}(GC, CGC) + \beta(\mathcal{G}_1) + \beta(\mathcal{G}_3) + \beta(\mathcal{G}_2)$. Each motif-matching region involves substrings that are in the language described by a context-free grammar in a given set \mathcal{C} . In this example, each grammar describes a stem-loop in the secondary *RNA* structure. In parts (a) and (b) we show parts of the *RNA* secondary structures implied by the alignment in Part (c).

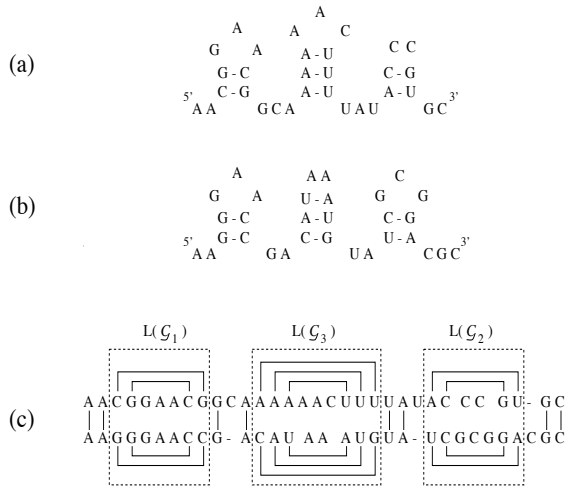


Fig. 1. Syntectic *RNA* sequences $S_1 = AACGGAACGGCAAAAACUUUUUAUACCCGUGC$ and $S_2 = AAGGGAACCGACAUAUAUGUAUCGCGGACGC$. 5' and 3' indicate the start and end of the sequences, respectively. Parts (a) and (b) respectively for S_1 and S_2 , are parts of the secondary structures implied by the alignment in Part (c).

For this example, $\mathcal{C} = \{\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3\}$. $\mathcal{G}_1 = (V, T, P, V_0)$ is a context free grammar, where $V = \{V_0, V_1, V_2, V_3\}$ is the set of variables, V_0 is the start variable, $T = \{A, C, G, U\}$ is the set of terminal symbols, and P includes the following production rules: $V_0 \rightarrow CV_1G \mid GV_1C$, $V_1 \rightarrow GV_2C$, $V_2 \rightarrow GAA$. \mathcal{G}_2 , and \mathcal{G}_3 are the following context free grammars: $\mathcal{G}_2 = \{\{V_0, V_1, V_2\}, \{A, C, G, U\}, \{V_0 \rightarrow AV_1U \mid UV_1A, V_1 \rightarrow CV_2G, V_2 \rightarrow CC \mid GCG\}, V_0\}$, and $\mathcal{G}_3 = \{\{V_0, V_1, V_2, V_3\}, \{A, C, G, U\}, \{V_0 \rightarrow AV_1U \mid GV_1C \mid CV_1G, V_1 \rightarrow AV_2U, V_2 \rightarrow AV_3U \mid UV_3A, V_3 \rightarrow AAC \mid AA\}, V_0\}$.

We present a dynamic programming algorithm for the *CFG-PSA* problem by modifying the classical dynamic programming solution for the global sequence alignment problem [31]. Our solution requires a solution for the following problem: given a string S of length n , and a *CFG* \mathcal{G}' , find all substrings of S generated by \mathcal{G}' where \mathcal{G}' is in Chomsky Normal Form. We modify the well-known *CYK* parsing algorithm [9], [18], [32] to solve this problem. This takes $O(n^3)$ time. Our dynamic programming solution uses this algorithm and collects all motif-matching substrings in each sequence in sets, and manipulates these sets in dynamic programming computations for solving the *CFG-PSA* problem. We assume that the context free grammars given in set \mathcal{C} are in Chomsky Normal Form. Our algorithm takes $O(|\mathcal{C}|n^3 + K_1K_2)$ time, where $|\mathcal{C}|$ is the number of grammars in set \mathcal{C} , $O(n)$ is the common length of the sequences aligned, and K_1 , and K_2 are the number of motif-matching substrings in the two sequences aligned respectively. The space requirement of our algorithm is $O(n^2 + K_1K_2)$.

We organize this paper as follows: in Section II, we summarize previous related work, and outline the motivation for the definition of the *CFG-PSA* problem. In Section III, we present an algorithm to parse all substrings of a given string. In Section IV, we present an algorithm for the *CFG-PSA* problem. We include pointers for possible extensions, and our remarks in Section V. We summarize our results in Section VI.

II. PREVIOUS RELATED WORK

There are many proposed models for *RNA* sequence, and structure comparisons: we can use extended edit operations (e.g. base-pair bond breaking), or represent *RNA* secondary structures by trees, and then compute tree-alignment. A comprehensive coverage of these techniques is given by Wang (Chap. 4 in [30]). Since base-pairing interactions can be described by context free grammars, there have been studies in the literature that involve *CFG*'s in *RNA* sequence alignment ([13], [8], [12]). Stochastic context free grammars (*SCFG*)s have been used in the description of *RNA* secondary structures [13]. An *SCFG* is a *CFG* where productions have probabilities. An *SCFG*-based method for sequence alignment was introduced by Eddy and Durbin [15], and by Sakakibara et al. [24]. Similar algorithms were developed by Lefebvre [20], [21]. Corpet and Michot [8] proposed an algorithm that aligns a given sequence of length n to a group of homologous

sequences of length m (i.e. m is length of the profile of these sequences) that have a common secondary structure, which is simply the set of paired positions. This algorithm computes an optimal score which is derived from both sequence, and structure similarities. It runs in $O(m^2n^3)$ time, and $O(m^2n^2)$ space. Corpet and Michot [8] also described a heuristic method for the problem. Notredame et al. [23] used the similarity definition used by Corpet and Michot [8], and proposed a genetic algorithm to align two *RNA* sequences when the secondary structure of one *RNA* is known. The main difference between these approaches and our approach with the definition of *CFG-PSA* problem is that in these approaches, a given sequence is aligned to another sequence (or a sequence profile) whose base-pairing positions are known and fixed. In *CFG-PSA* problem base-pairing positions are not fixed in any of the sequences; there may be many candidate pairwise matches for common (sub)structures (determined by the given set of motifs) in both sequences; one that yields the optimum score is chosen. In short, in our approach the alignment process has complete freedom on both sequences, while in these approaches the freedom is limited to one sequence only.

Dowell and Eddy [12] use *pair stochastic context-free grammars*, *pairSCFGs*, for pairwise structural alignment of *RNA* sequences with the constraint that the structures are confidently aligned at positions (pins) which are known a priori. The main difference from our work here is that Dowell and Eddy [12] assume a general *SCFG* that works well for all *RNA* sequences, whereas we use the knowledge of known set of (weighted) motifs to guide the alignment process.

The regular-expression constrained sequence alignment problem defined by Arslan [1], and the work of Comet and Henry [10] have some similarities with our work in this paper. However, in this paper our definition of a motif is more general, and this gives rise to a new constraint sequence alignment problem whose solution involves using additional techniques. Constraining alignments to contain common motifs [1], or rewarding inclusion of common motifs in alignments [10] have been proven to yield biologically more relevant alignments.

Homologous *RNAs* tend to conserve a common base-paired secondary structure motifs that can be expressed by a context free grammar. In our definition of the *CFG-PSA* problem, we use *CFGs* to describe motifs. An input to the problem is a set of such *CFGs*, and we use the classical Needleman-Wunsch global alignment algorithm [22], and reward each motif-matching region by a score assigned to each motif by a given function. We explain the motivation for these as follows:

- Context free grammars can be used for sequence motifs that are strings, or regular expressions (e.g. PROSITE patterns). They can also describe more complex motifs. For example, base-pairings in stem-loops in *RNA* secondary structure can be described by a context free grammar.
- Motifs that are used in guiding an alignment sought may come from a database. Therefore, in our formulation we take a set \mathcal{C} of motifs as input. For regular-expression motifs PROSITE (<http://www.expasy.org/prosite>) has been a database of sequence motifs, which were represented and

stored as regular expressions. It is essential to compile a comprehensive library of structural motifs identifiable in sequences. A large number of *RNA* motifs that bind proteins are already known [4]. New motifs continue to be discovered [19].

- Sequences aligned may have many motifs in common. Comet and Henry [10], and Arslan [2] considered alignments that contain multiple PROSITE patterns. Alignments may contain varying number of motifs in different orders. Some motifs may occur in an *RNA* sequence multiple times. For example, the *K-turn* motif occurs six times in *H.marismortui* 23*S RNA*, and twice in *T.thermophilus* 16*S RNA*. This motif can appear in variations each of which can be described by a *CFG*. Since the class of context free grammars is closed under the union operation ([17]), this motif can be described by a *CFG* that captures all these variations. Discovery of any sequence described by this *CFG* in an *RNA* sequence should be predictive of an occurrence of a *K-turn* motif. The purpose of the alignment process is not only to find an alignment that includes motif-matches, but also to include all or some of them in an order to optimize the resulting score.
- Constrained sequence alignment problems [1], [3], [26], [27] require that alignments sought are required to contain a given motif. Comet and Henry [10] rewards alignments that contain common motifs. We follow this latter approach in this case because if rewarding mechanism is properly set, it constrains the alignments to contain common motifs, and it makes the problem easier when there are multiple common motifs.
- We assume that a given function assigns for each motif-match a score fixed for the motif involved. The method of Comet and Henry [10] aligns each motif-matching region, and adds to its score x , additional score that is the product of x and a given reward-coefficient for the motif involved. The definition given for regular-expression constrained sequence alignment by Arslan [1] requires that the alignment scores are computed for motif-matching regions, and used in computing the total alignment score. We believe that alignment of these regions unnecessarily complicate the problem when we align *RNA* sequences since these regions are not necessarily conserved. We can set a fixed score for all motif-matches that involve variations of a given motif. This yields an efficient dynamic programming solution that we present in Section IV. Such an efficient solution is unlikely to be found if we follow the rewarding approach in [10] when there are multiple common motifs.
- For the alignment of non-motif regions we propose using the Needleman-Wunsch algorithm [22] for simplicity. Other more advanced algorithms [13] can also be used.

III. ALL SUBSTRING PARSING

A *context-free grammar* is a quadruple $G = (V, T, P, V_0)$, where V is the set of variables, T is the set of terminal

symbols, P is the set of production rules, and $V_0 \in V$ is the start variable. A production is of the form $X \rightarrow \alpha$, where $X \in V$, and $\alpha \in (V \cup T)^*$. We use $X \rightarrow \alpha_1 \mid \dots \mid \alpha_r$ to denote the productions $X \rightarrow \alpha_1, \dots, X \rightarrow \alpha_r$.

We use uppercase letters to denote variables, and lower case letters to denote terminal symbols. Greek lower case letters denote sentential forms, i.e. strings in $(V \cup T)^*$.

Let $L(\mathcal{G})$ be the set of strings that are generated by \mathcal{G} using the productions in P starting with the start variable V_0 .

A context-free grammar is in Chomsky Normal Form if every rule is of the form $X \rightarrow YZ$, and $X \rightarrow a$ where a is any terminal and X, Y , and Z are any variables except that neither Y nor Z is the start variable. $V_0 \rightarrow \epsilon$ can be a production, where V_0 is the start variable, and ϵ is the null-string. Any given $CFG \mathcal{G}$ can be converted to an equivalent $CFG \mathcal{G}'$ in Chomsky Normal Form, i.e. $L(\mathcal{G}') = L(\mathcal{G})$. The proof of this theorem, and many properties of context free languages can be found in [17], [25]. For the $CFG \mathcal{G}_1$ that we use in Figure 1, an equivalent CFG in Chomsky Normal Form is $\mathcal{G}'_1 = (\{V_0, V_1, V_2, V_3, V_4, V_6, V_8, V_9\}, \{A, C, G, U\}, P', V_0)$, where P' includes the following production rules: $V_0 \rightarrow CV_3 \mid GV_4$, $V_1 \rightarrow GV_5$, $V_2 \rightarrow GV_6$, $V_3 \rightarrow V_1V_7$, $V_4 \rightarrow V_1V_8$, $V_5 \rightarrow V_2V_8$, $V_6 \rightarrow AV_9$, $V_7 \rightarrow C$, $V_8 \rightarrow C$, $V_9 \rightarrow A$.

Parsing is the problem of finding a sequence of productions in P that derives a given string from the start variable V_0 . There are numerous parsing algorithms. On general context-free grammars, all practical parsing algorithms run in $O(n^3)$ time [14], [9], [18], [32]. There is a complicated algorithm by Valiant [29] which runs in $O(n^{2.81})$. There are faster algorithms for subclasses (such as unambiguous grammars) of context-free grammars [14].

Lemma 1: Given a string S , and a $CFG \mathcal{G}$ in Chomsky Normal Form, we can find all substrings of S that are generated by \mathcal{G} in $O(n^3)$ time, where n is the length of S .

Proof: We prove this lemma by presenting an algorithm that finds all substrings of S that are generated by \mathcal{G} in $O(n^3)$ time.

There is a parsing algorithm known as the Cocke-Younger-Kasami (CYK) algorithm [9], [18], [32]. It is a dynamic programming algorithm that runs on any CFG . We modify this algorithm to find for any given string S all substrings that are in $L(\mathcal{G})$ for a given $CFG \mathcal{G}$, where \mathcal{G} is in Chomsky Normal Form. We present this algorithm in Figure 2.

The algorithm uses dynamic programming to find substrings derived from variables. The lengths of these substrings grow iteratively. In Step 1, each position i in S is examined to see if there is a variable X that produces $S[i]$. All such variables are added to $T[i, i]$. Then iteratively substrings of lengths $2, 3, \dots, n$ are considered in increasing length in Step 2. For each length, substrings $S[i..j]$ of this length are considered. Every rule of the form $X \rightarrow YZ$ is added to $T[i, j]$ if there exists k such that $Y \in T[i, k]$ and $Z \in T[k + 1, j]$. Step 3 collects all substring start-end positions (i, j) in P where the start variable V_0 derives $S[i..j]$, i.e. $S[i..j]$ is in $L(\mathcal{G})$. ■

Algorithm ParseAll
input string S , length n , $CFG \mathcal{G}$

Step 1. Find the substrings derived by the rules of the form $X \rightarrow a$:

```
for  $i = 1$  to  $n$  do
  set  $T[i, i] = \emptyset$ 
  for each variable  $X$ 
    if  $X \rightarrow S[i]$  is a rule then
      add  $X$  to  $T[i, i]$ 
```

Step 2. Find the substrings derived by the rules of the form $X \rightarrow YZ$:

```
for  $l = 2$  to  $n$  do
  for  $i = 1$  to  $n - l + 1$  do
    set  $j = i + l - 1$ 
    for  $k = i$  to  $j - 1$  do
      for each rule  $X \rightarrow YZ$ 
        if  $Y \in T[i, k]$  and  $Z \in T[k + 1, j]$  then
          add  $X$  to  $T[i, j]$ 
```

Step 3. Return the set of all substrings generated by \mathcal{G} :

```
 $P = \emptyset$ 
for  $i = 1$  to  $n$  do
  for  $j = i$  to  $n$  do
    if the start variable  $V_0 \in T[i, j]$  then
      add  $(i, j)$  to  $P$ 
return  $P$ 
```

Fig. 2. Algorithm ParseAll.

IV. CFG-PSA ALGORITHM

We denote by $S[i..j]$ the substring $s_i s_{i+1} \dots s_j$ of string $S = s_1 s_2 \dots s_n$. If $i > j$ then $S[i..j]$ is the null string.

The objective of sequence alignment is to quantify the similarity between two given strings $S_1[1..n_1]$ and $S_2[1..n_2]$ under a given *scoring scheme*. The following is the classical dynamic programming formulation [31] that computes the maximum global alignment score for S_1 and S_2 under the *simple similarity scheme* where a given function γ assigns a score for each alignment column:

$$\mathcal{H}_{i,j} = \max \left\{ \begin{array}{l} \mathcal{H}_{i-1,j} + \gamma(S_1[i], '-'), \\ \mathcal{H}_{i-1,j-1} + \gamma(S_1[i], S_2[j]), \\ \mathcal{H}_{i,j-1} + \gamma('-', S_2[j]) \end{array} \right\} \quad (1)$$

for all i, j , $1 \leq i \leq n_1$, $1 \leq j \leq n_2$, with the boundary values $\mathcal{H}_{0,0} = 0$, $\mathcal{H}_{i,-1} = \mathcal{H}_{-1,j} = -\infty$. Then \mathcal{H}_{n_1,n_2} is the maximum global alignment score between S_1 and S_2 . The maximum global alignment score can be computed in time $O(n_1 n_2)$ using $O(\min\{n_1, n_2\})$ space because only $O(\min\{n_1, n_2\})$ entries of the dynamic programming matrix need to be stored at any given time [31].

Let X_k be the set of all (i', i) such that $S_1[i'..i]$ is generated by \mathcal{G}_k . Similarly, let Y_k be the set of all (j', j) such that $S_2[j'..j]$ is generated by \mathcal{G}_k . The following is a dynamic programming solution for the CFG -PSA problem:

$$\begin{aligned}
\mathcal{H}_{i,j} = & \\
\max\{ & \mathcal{H}_{i-1,j} + \gamma(S_1[i], ' - '), \\
& \mathcal{H}_{i-1,j-1} + \gamma(S_1[i], S_2[j]), \\
& \mathcal{H}_{i,j-1} + \gamma(' - ', S_2[j]), \\
& \max_{(i',i) \in X_k, (j',j) \in Y_k, \mathcal{G}_k \in \mathcal{C}} \{ \mathcal{H}_{i',j'} + \beta(\mathcal{G}_k) \} \} & (2)
\end{aligned}$$

with the same boundary conditions: for all i, j , $1 \leq i \leq n_1$, $1 \leq j \leq n_2$, with the boundary values $\mathcal{H}_{0,0} = 0$, $\mathcal{H}_{i,-1} = \mathcal{H}_{-1,j} = -\infty$.

For the additional max-term $\max_{(i',i) \in X_k, (j',j) \in Y_k} \{ \mathcal{H}_{i',j'} + \beta(\mathcal{G}_k) \}$ at (i, j) we consider all (i', i) in X_k and (j', j) in Y_k if both X_k and Y_k are not empty. We note that each such (i', i) - (j', j) -pair implies that there exists a motif-match of the motif described by the CFG \mathcal{G}_k . That is, $S_1[i'..i] \in L(\mathcal{G}_k)$, and $S_2[j'..j] \in L(\mathcal{G}_k)$. Including this motif-match in an alignment yields an additional score $\mathcal{H}_{i',j'} + \beta(\mathcal{G}_k)$. We consider all possible motif-matches at (i, j) , and compute the maximum score.

We claim that $\mathcal{H}_{i,j}$ computed this way is the optimum CFG-PSA score of $S_1[1..i]$ and $S_2[1..j]$. For the correctness proof we can use induction. The induction is on the indices i, j such that when $\mathcal{H}_{i,j}$ is computed $\mathcal{H}_{i,j-1}$, $\mathcal{H}_{i-1,j-1}$, and $\mathcal{H}_{i-1,j}$ have all been computed. Base cases are correct due to the boundary values. Two nested loops generating i (the outer loop), and j (the inner loop) can guarantee a correct ordering for the induction. It is easy to see that when $\mathcal{H}_{i,j}$ is computed this way our induction hypothesis will be correct.

We examine all motif-matches in X_k , and Y_k . We compute X_k , and Y_k for all $\mathcal{G}_k \in \mathcal{C}$ as follows: for all \mathcal{G}_k , we set $X_k = \text{ParseAll}(S_1, n_1, \mathcal{G}_k)$, and $Y_k = \text{ParseAll}(S_2, n_2, \mathcal{G}_k)$. Then we collect all X_k into X , and Y_k into Y , and sort and process them in ascending order of their end-points. We move every (i', i) in X_k to X as (i', i, k) . Similarly, we move every (j', j) in Y_k to Y as (j', j, k) . We sort each list X , and Y in ascending order of the end-points i, j . We compute $\mathcal{H}_{i,j}$ in the order described in our induction. In computing $\mathcal{H}_{i,j}$, we collect all (i', i, k) into X_i if such (i', i, k) 's exist (since X_i is sorted, if they exist, the current list pointer points to the first one) and advance the pointer to the next end-point. Similarly, we copy all (j', j, k) into Y_j if such (j', j, k) 's exist (if they exist the current list pointer points to the first one) and advance the pointer cyclically to the next end-point (the pointer is advanced to the beginning of the list Y after the entire list Y is processed). Next, we compute the cartesian product $X_i \times Y_j$. Then the additional max-term can be computed as follows:

$$\max_{(i',i,k_1,j',j,k_2) \in X_i \times Y_j, k_1=k_2} \{ \mathcal{H}_{i',j'} + \beta(\mathcal{G}_{k_1}) \}$$

Computing each X_k takes $O(n_1^3)$ time, and each Y_k takes $O(n_2^3)$ time. Computing all X_k , and all Y_k , and collecting them into X , and Y respectively takes $O(|\mathcal{C}|n_1^3 + |\mathcal{C}|n_2^3)$ time, where $|\mathcal{C}|$ is the number of motifs (CFGs) in \mathcal{C} . Let K_1 , and K_2 be the number of motifs that appear in S_1 , and S_2 , respectively. That is, $K_1 = |X|$, and $K_2 = |Y|$. Sorting X , and Y takes $O(K_1 \log K_1 + K_2 \log K_2)$ time. The total

time spent on computing the additional max-terms (the last term in Equation (2)) for the entire dynamic programming computations is $O(K_1 K_2)$. Putting all together, the modified dynamic programming takes $O(|\mathcal{C}|n_1^3 + |\mathcal{C}|n_2^3 + n_1 n_2 + K_1 K_2)$ time.

If we want to find an optimal alignment, we can run this algorithm to find motif-matching positions in S_1 , and in S_2 . Then we can use Hirschberg's algorithm [16] to align the remaining segment-pairs in S_1 and S_2 . The total space requirement for ordinary sequence alignments is $O(\min\{n_1, n_2\})$. But, *ParseAll* requires $O(n_1^2 + n_2^2)$ space. We also need $O(K_1 K_2)$ space to compute and store the cartesian product $X \times Y$. Therefore, the total space complexity is $O(n_1^2 + n_2^2 + K_1 K_2)$.

V. POSSIBLE EXTENSIONS & REMARKS

Our CFG-PSA algorithm can be modified to run for *affine gap* penalties, and also for local alignment computations. These cases have dynamic programming formulations similar to (1). They can be used in the alignment of non-motif-matching regions. These replacements do not change the complexity of our algorithm.

We can also use other more advanced algorithms [13], [12], [8] in these parts.

We can verify that our definition, and algorithm for CFG-PSA can be generalized to the problem of optimally aligning multiple sequences guided by a given set of motifs described by CFGs. Also, the *progressive sequence alignment technique* [13] used for other constrained sequence alignment problems [26], [7] can be used for this case. Progressive alignment is a heuristic that does not always guarantee an optimal alignment. It first picks a pair of sequences, and aligns them. Then, a third sequence is chosen and aligned to the first alignment. This process continues until all sequences are aligned.

VI. CONCLUSION

We introduce the context-free grammars guided pairwise sequence alignment problem. Our algorithm finds in each sequence all substrings that are generated by a given set of context free grammars. Each of these grammars describe a motif. A given function assigns a score for each motif-matching region in an alignment. The total score of an alignment is the total score of motif-matching regions, and the total score of the ordinary pairwise alignments of regions outside and in between these motif-matching regions. We present a dynamic programming algorithm that finds an optimal alignment for this problem. Our algorithm can be adapted for affine gap penalties, and also for computing local alignments. Our definition, and algorithm can also be generalized for the alignment of multiple sequences guided by context free grammar-described-motifs.

REFERENCES

- [1] A. N. Arslan. Regular expression constrained sequence alignment. *Journal of Discrete Algorithms*, Elsevier, (published online: <http://dx.doi.org/10.1016/j.jda.2007.01.003>) (in press) (an earlier version appeared in: *CPM 2005, Jeju Island, Korea, June 19-22, 2005, LNCS 3537*, pp. 322-333, 2005)

- [2] A. N. Arslan. Multiple sequence alignment containing a sequence of regular expressions. *IEEE 5th Symposium on Intelligent Computations on Bioinformatics and Biotechnology (CIBCB) 2005, San Diego, November 14-15*, pp.230-236, 2005.
- [3] A. N. Arslan and Ö. Egecioglu. Algorithms for the constrained common sequence problem. *Int. J. of Found. of Comp. Sci.*, 16(6): pp. 1099–1109, 2005.
- [4] C. G. Burd and G. Dreyfuss. Conserved structures and diversity of functions of *RNA* binding motifs. *Science*, 265:615-621, 1994.
- [5] F. Y. L. Chin, N. L. Ho, T. W. Lam, P. W. H. Wong, M. Y. Chan. Efficient constrained multiple sequence alignment with performance guarantee. *Proc. IEEE Computational Systems Bioinformatics (CSB 2003)*, pp. 337-346, 2003.
- [6] F. Y.L. Chin, A. D. Santis, A. L. Ferrara, N. L. Ho, S. K. Kim. A simple algorithm for the constrained sequence problems. *IPL*, 90:175-179, 2004.
- [7] Chung, Y.-S., Lee, W.-H., Tang, C. Y., and Lu, C. L. RE-MuSiC: a tool for multiple sequence alignment with regular expression constraints. *Nucleic Acids Research* (in press) (online: doi:10.1093/nar/gkm275)
- [8] F. Corpet and B. Michot. RNAalign program: alignment of RNA sequences using both primary and secondary structures. *Computers Applications in the Biosciences*, 10: 389–399, 1994.
- [9] J. Cocke and J. T. Schwartz. Programming languages and their compilers: Preliminary notes. *Technical report*, Courant Institute of Mathematical Sciences, New York University, 1970.
- [10] J.-P. Comet and J. Henry. Pairwise sequence alignment using a PROSITE pattern-derived similarity score. *Comp. and Chem.*, 26, pp. 421-436, 2002.
- [11] R. F. Doolittle. Similar amino acid sequences: chance or common ancestry. *Science*, 214:149-159, 1981.
- [12] R. D. Dowell and S. R. Eddy. Efficient pairwise *RNA* structure prediction and alignment using sequence alignment constraints. *BMC Bioinformatics*,7(400):1-18, 2006.
- [13] R. Durbin, S. Eddy, A. Krogh, and G. Michison. *Biological sequence analysis*. Cambridge University Press, 1998.
- [14] J. Early. An efficient context-free parsing algorithm. *Comm. ACM*, 13:2, 94-102, 1970.
- [15] S. R. Eddy and R. Durbin. *RNA* sequence analysis using covariance models. *Nucleic Acid Research*, 22:2079-2088, 1994.
- [16] D.S. Hirschberg. Algorithms for the longest common subsequence problem. *J. ACM*, 24:664–675, 1977.
- [17] J. E. Hopcroft and J. D. Ullman. Introduction to automata theory, languages, and computation. *Addison-Wesley Publishing Company*, 1979.
- [18] T. Kasami. An efficient recognition and syntax-analysis algorithm for context-free languages. *Scientific report AFCRL-65-758*, Air Force Cambridge Research Lab, Bedford, MA, 1965.
- [19] D. J. Klein, T. M. Schmeing, P. B. Moore, and T. A. Steitz. The kink-turn: a new *RNA* secondary structure motif. *The EMBO Journal*, 20(15):4214-4221, 2001.
- [20] F. Lefebvre. An optimized parsing algorithm well suited to *RNA* folding. In C. Rawlings, D. Clark, R. Altman, L. Hunter, T. Lengauer, S. Wodak, eds., *Proc. of the Third Int. Conf. on Intelligent Systems for Molecular Biology*, 222-230, AAAI Press, 1995.
- [21] F. Lefebvre. A grammar-based unification of several alignment and folding algorithms. In D. J. States, P. Agarwal, T. Gaasterland, L. Hunter, and R. F. Smith, eds., *Proceedings of the Fourth International Conference on Intelligent Systems for Molecular Biology*, 143-154, AAAI Press, 1996.
- [22] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.*, 48:443-453, 1970.
- [23] C. Notredame, E. A. O'Brien and D. G. Higgins. RAGA: RNA sequence alignment by genetic algorithm. *Nuc. Acids Res.*, 25(22):4570–4580, 1997.
- [24] Y. Sakakibara, M. Brown, R. Hughey, I. S. Mian, K. Sjölander, R. C. Underwood, and D. Haussler. Stochastic context-free grammars for *tRNA* modeling. *Nucleic Acid Research*, 22:5112-5120, 1994.
- [25] Introduction to the theory of computation, second edition. *Thomson Course Tech.*, Boston, MA, 2006.
- [26] C. Y. Tang, C. L. Lu, M. D.-T. Chang, Y.-T. Tsai, Y.-J. Sun, K.-M. Chao, J.-M. Chang, Y.-H. Chiou, C.-M. Wu, H.-T. Chang, and W.-I. Chou. Constrained multiple sequence alignment tool development and its applications to rase family alignment. *Proceeding of the 1st IEEE Computer Society Bioinformatics Conference (CSB 2002)*, pp. 127-137, 2002.
- [27] Y.-T. Tsai. The constrained common sequence problem. *IPL*, 88:173–176, 2003.
- [28] Y.-T. Tsai, C. L. Lu, C. T. Yu, and Y. P. Huang. MuSiC: A tool for multiple sequence alignment with constraint. *Bioinformatics*, 20(14):2309-2311, 2004.
- [29] L. Valiant. General context free recognition in less than cubic time. *J. Computer and System Sciences*, 10(2):308-315, 1975.
- [30] J. T. L. Wang, M. J. Zaki, H. T. T. Toivonen, and D. Shasha (Eds). *Data mining in bioinformatics*. Springer, London, 2005.
- [31] M. S. Waterman. *Introduction to computational biology*. Chapman & Hall, 1995.
- [32] D. H. Younger. Recognition and parsing of context-free languages in time n^3 . *Information and Control*, 10(2): 189208, 1967.