IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 23, 2018, accepted April 20, 2018, date of publication May 2, 2018, date of current version May 24, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2832209

NETASPNO: Approximate Strict Pattern Matching Under Nonoverlapping Condition

YOUXI WU[1,2,3], (Member, IEEE), SHASHA LI[1,2], JINGYU LIU[1,2], LEI GUO[4], AND XINDONG WU[5], (Fellow, IEEE)

[1]School of Computer Science and Engineering, Hebei University of Technology, Tianjin 300401, China
[2]Hebei Province Key Laboratory of Big Data Calculation, Tianjin 300401, China
[3]Engineering Research Center of Intelligent Rehabilitation and Detecting Technology, Ministry of Education, Tianjin 300131, China
[4]School of Electrical Engineering, Hebei University of Technology, Tianjin 300131, China
[5]School of Computing and Informatics, University of Louisiana at Lafayette, Lafayette, LA 70504, USA

Corresponding author: Youxi Wu (wuc567@163.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61673159 and Grant 61571180 and in part by the U.S. National Science Foundation under Grant IIS-1613950.

ABSTRACT In pattern matching, a gap constraint is a more flexible wildcard than traditional wildcards "?" and "*". Pattern matching with gap constraints is more difficult to handle and fulfills user's enquiries more easily. Pattern matching with gap constraints has therefore been carried out in numerous research works, such as music information retrieval, searching protein sites, and sequence pattern mining. Strict pattern matching under a nonoverlapping condition, as a type of pattern matching with gap constraints, is a key issue of sequence pattern mining with gap constraints since it can be used to compute the frequency of a pattern. Exact matching limits the flexibility of the match to some extent since it requires each character to be matched exactly. We therefore address approximate strict pattern matching under the nonoverlapping constraints (ASPNO) and propose an effective algorithm, named NETtree for ASPNO (NETASPNO), which first transforms the problem into a Nettree data structure, an extensive tree structure. To find the nonoverlapping occurrences effectively, we propose the concept of number of roots paths with distance constraints (NRPDC) which indicates the number of path from a node to the roots with distance d and can be used to delete useless parent-child relationships and useless nodes. We iteratively recalculate the NRPDCs of each node on the subnettree with the rightmost root. Then we can get a path from the rightmost leaf to its rightmost root without using the backtracking strategy. NETASPNO therefore iteratively gets the rightmost root-leaf-path and prunes the path on the Nettree. Extensive experimental results demonstrate that NETASPNO has better performance than the other competitive algorithms.

INDEX TERMS Approximate pattern matching, wildcard, gap constraint, sequence, occurrence.

I. INTRODUCTION

Pattern matching (or string matching) has played a very important role in many research fields [1], [2]. Numerous research works have been carried out on this task, such as network intrusion detection systems [3], approximate string search in large scale string [4] or in large spatial databases [5], text indexing [6], pattern queries on XML data [7], and document retrieval [8].

One of the essential tasks in pattern mining is to calculate the support of a pattern, which can be seen as a pattern matching task [9]. Therefore, pattern matching is one of the essential tasks in pattern mining. For instance, Chen et al. [10] focused on exact circular string matching. Based on this technology, circular pattern discovery was proposed [11].

Recently, many research works have focused on pattern matching with gap constraints (or flexible gaps or flexible wildcards) [12]–[14], which is a kind of wildcard that is more flexible than the traditional wildcards "?"and "*". For example, in computational biology, protein pattern matching employed this type of pattern matching to find some special protein sites [15]. RNA structure can also be found based on pattern matching with flexible gaps [16]. As mentioned above, pattern matching with gap constraints is also one of the essential tasks in sequence pattern mining. Numerous research works have been proposed to mine the patterns with gap constraints, which are applied in many fields, such as time series analysis [17], medical emergency identification [18], customer purchase

patterns mining [19], biological characteristics mining [20], and feature selection for sequence classification [21]. A gap constraint can be written as "a[x, y]b", where 'a' and 'b' are two characters and x and y are two integer numbers that represent the minimal and maximal numbers of any characters [22]. For instance, both subsequences "ACCT" and "AGGCT" are two occurrences of pattern "A[2,3]T" since the first and the last characters of subsequences are 'A' and 'T', respectively, and there are two or three characters between 'A' and 'T' in the subsequences, respectively. Pattern P with gap constraints [23], [24] can be written as $p_1[min_1, max_1]p_2 \cdots [min_j, max_j]p_{j+1} \cdots [min_{m-1}, max_{m-1}]p_m$. Pattern matching with gap constraints can fulfil user enquiries more easily and is more flexible. But this issue is more difficult to solve and various versions have been investigated, such as the traditional pattern matching and the strict pattern matching. The strict pattern matching, unlike the traditional pattern matching which uses the last position in the sequence to describe an occurrence, employs a group of positions in the sequence to express an occurrence. Apparently, the strict pattern matching considers the matching process in detail, while the traditional pattern matching ignores the process. Pattern matching under the nonoverlapping condition [25], as a kind of strict pattern matching with gap constraints, means that each subsequence can be used no more than once by each subpattern and this matching method has been applied in sequence pattern mining with gap constraints [26], [27]. An illustrative example of pattern matching under the nonoverlapping condition is shown as follows.

Example 1: We have sequence $S = s_1s_2s_3s_4s_5s_6 =$ *ABBABA and pattern* $P = p_1[min_1, max_1]p_2[min_2, max_2]p_3 =$ *A[0,1]B[0,1]A.*

According to subpattern A[0,1]B, there is no wildcard or one wildcard "?" between 'A' and 'B'. For example, we can see that both s_1s_2 *and* s_1s_3 *are two suboccurrences of subpattern A[0,1]B. Therefore, suboccurrence* s_1s_2 *can be expressed by* $\langle 1,2\rangle$*. It is easy to see that in this example there are three occurrences:* $\langle 1,2,4\rangle$*,* $\langle 1,3,4\rangle$*, and* $\langle 4,5,6\rangle$*. Although* s_4 *is used in both* $\langle 1,2,4\rangle$ *and* $\langle 4,5,6\rangle$*, it matches* p_3 *and* p_1*, respectively. Hence,* $\langle 1,2,4\rangle$ *and* $\langle 4,5,6\rangle$ *are called two nonoverlapping occurrences [26], [27]. However,* $\langle 1,2,4\rangle$ *and* $\langle 1,3,4\rangle$ *are two overlapping occurrences because* s_1 *matches* p_1 *twice in the two occurrences.*

As we know, exact matching requires each character to be matched exactly. However, in many cases, exact matching is not good enough to satisfy certain criteria, since exact matching limits the flexibility of the matching to some extent. The approximate version should therefore be considered [28], [29].

Example 2: We also use the same sequence S and pattern P as in Example 1.

Under exact matching $\langle 1,2,3\rangle$ *is not an occurrence for pattern P, because* $s_3 = B \neq p_3 = A$*, but it is an occurrence of approximate pattern matching with the Hamming distance 1.* $\langle 1,2,4\rangle$ *is also an occurrence of approximate pattern matching with the Hamming distance 1, since the Hamming distance between* $s_1s_2s_4$ *and* $p_1p_2p_3$ *is 0 which is smaller than 1. Hence, all exact occurrences are special cases of approximate pattern matching with the Hamming distance. With the Hamming distance 1,* $\langle 3,4,5\rangle$ *is not an occurrence, since the Hamming distance between* $s_3s_4s_5$ *and* $p_1p_2p_3$ *is 3 which is greater than 1. From this example we can see that if the similarity constraint is 0, the approximate pattern matching automatically converts to exact pattern matching. Therefore, compared with exact pattern matching, approximate matching is more general.*

To tackle this challenge, first we transform the approximate strict pattern matching under nonoverlapping condition problem into a Nettree data structure which is an extensive tree structure. We also propose a concept to indicate the number of paths from a node to its roots with distance d. Using this concept, we can delete useless parent-child relationships and useless nodes effectively. We also reuse this concept on the subnettree with the rightmost root and iteratively get a path from the rightmost leaf to its rightmost root without using the backtracking strategy. At last, we iteratively get the rightmost root-leaf-path and prunes the path on the Nettree. The contributions of this paper are threefold:

(1) Due to the limitation of the exact pattern matching, in this paper, we formally address Approximate Strict Pattern matching under the NonOverlapping condition (ASPNO) which is a more general version than the exact version.

(2) We propose an effective algorithm, named NETtree for ASPNO (NETASPNO), which iterates to find the rightmost root-leaf path. The space and time complexities are $O(n*m*(T+g))$ and $O(n*m^2*T*g)$ in the worst case, where n, m, g, and T are the length of sequence, the length of pattern, the maximal gap, and the similarity constraint, respectively.

(3) Experimental results show that NETASPNO has better performance than other algorithms.

The rest of this paper is organized as follows. Section 2 summarizes the related work. Section 3 presents the definition of the problem, proposes the algorithm NETASPNO, and shows the time and space complexities of NETASPNO. Section 4 validates the performance of the algorithm. We present the conclusion in Section 5.

II. RELATED WORKS

The gap constraints make the issue not only more difficult but also more flexible. There are two kinds of pattern matching with gap constraints: traditional pattern matching and strict pattern matching [28]. Under the traditional pattern matching, an occurrence is the last position of a matching in the sequence, while under the strict pattern matching an occurrence is a group of positions of each subpattern in the sequence. Both the traditional pattern matching and the strict pattern matching have also been widely applied in many tasks. For example, Navarro and Raffinot [15] proposed an effective algorithm that employed the traditional pattern matching with applications to protein

sites searching. Some kinds of sequence pattern mining tasks [20], [24], [26], [27], [30], [31] used the strict pattern matching strategies to calculate the support of a pattern. There are three types of sequence pattern mining conditions, i.e. no special condition [20]–[23], the one-off condition [30], [31], and the nonoverlapping condition [26], [27]. Moreover, there are also three types of the strict pattern matching strategies, strict pattern matching under no special condition [22], [28], under the one-off condition [32], [33], and under the nonoverlapping condition [25]. Now, Example 3 is employed to illustrate the relationship between the traditional pattern matching and the strict pattern matching under no special condition, under the one-off condition, and under the nonoverlapping condition.

Example 3: Suppose we have sequence $S = ABBABA$ *and pattern* $P = A[0, 1]B[0, 1]A$, *all occurrences are shown in Table 1.*

TABLE 1. All occurrences for pattern *P* in sequence *S*.

	1	2	3	4	5	6	
S=	A	B	B	A	B	A	
	A	B		A			The first occurrence
	A		B	A			The second occurrence
				A	B	A	The third occurrence

Under the traditional pattern matching, the last position in the sequence is considered. The three occurrences have two different last positions: 4 and 6. Hence, there are two occurrences under the traditional pattern matching. To describe an occurrence conveniently, a group of positions is used under the strict pattern matching. $\langle 1, 2, 4\rangle$, $\langle 1, 3, 4\rangle$, *and* $\langle 4, 5, 6\rangle$ *are the three occurrences for strict pattern matching under no special condition. Strict pattern matching under the one-off condition and under the nonoverlapping condition are of different subsets from strict pattern matching under no special condition.* $\langle 1, 2, 4\rangle$ *and* $\langle 4, 5, 6\rangle$ *are two occurrences under the nonoverlapping condition since each subsequence can be used to match different subpatterns while under the one-off condition, there is only one occurrence for this instance. For example, if* $\langle 1, 2, 4\rangle$ *is selected,* $\langle 4, 5, 6\rangle$ *cannot be selected since each subsequence can be used only once. The two problems, under the one-off condition and under the nonoverlapping condition, have different computational complexity since the former is a NP-hard problem [34] while the latter is a P problem [25]. For clarification, Table 2 reports the occurrences under the different methods.*

From this example, we can see that the strict pattern matching is more detailed to describe an occurrence but more difficult to handle than the traditional pattern matching. Table 3 shows a comparison of related works.

From Table 3, we can see that the most relevant related work is [25] which is an exact version of our problem. As we know exact matching is too tight to find the useful information, while approximate matching with certain criteria is good enough. Similarly, some useful patterns may be lost under exact sequence pattern mining. When the distance is 0, approximate pattern matching transforms into exact version automatically. Hence, approximate pattern matching is a more general issue. Due to the limitations of the exact version, we focus on ASPNO.

III. PROBLEM DEFINITIONS AND ALGORITHMS

A. PROBLEM DEFINITIONS

Definition 1: A sequence S with length n can be written as $s_1 s_2 \ldots s_i \ldots s_n (1 \le i \le n)$ *and* $s_i \in \sum$, *where* $\sum$ *is a set of characters. A pattern P with gap constraints can be written as* $p_1[min_1, max_1]p_2 \cdots [min_j, max_j]p_{j+1} \cdots [min_{m-1}, max_{m-1}]p_m (1 \le j \le m)$ *and* $p_j \in \sum$, *where m is the length of P and* min_j *and* max_j *are two given non-negative integers which refer to the minimal and maximal wildcards "?" between* p_j *and* p_{j+1}, *respectively.*

As we know, in a DNA sequence, $\sum$ is {A, C, G, T}.

Definition 2: Let $P = p_1 p_2 \ldots p_m$ *and* $Q = q_1 q_2 \ldots q_m$ *be two sequences with length m. The Hamming distance between P and Q denoted by* $D(P, Q)$ *is the number of positions at which the corresponding characters are different.*

Example 4: Suppose $P = p_1 p_2 p_3 = AGT$ *and* $Q = q_1 q_2 q_3 = ACT$ *are given. According to Definition 2, we can see that* p_2 *and* q_2 *are different. Therefore, the Hamming distance between P and Q is 1; that is,* $D(P, Q) = 1$.

Definition 3: Given a threshold T, if a group of position indexes $L = \langle l_1, l_2, \cdots l_m\rangle$ *is an approximate occurrence of pattern P in sequence S, L should satisfy the following equations.*

$$min_{j-1} \le l_j - l_{j-1} - 1 \le max_{j-1} \quad (1)$$

$$D(p_1 p_2 p_m, s_{l_1} s_{l_2} s_{l_m}) \le T \quad (2)$$

Definition 4: Let $L = \langle l_1, l_2, \ldots l_m\rangle$ *and* $L' = \langle l'_1, l'_2, \cdots l'_m\rangle$ *be two approximate occurrences of pattern P in sequence S. If and only if for all j* $(1 \le j \le m)$, l_j *is not equal to* l'_j, *that is* $l_j \ne l'_j$, *L and L′ are two nonoverlapping approximate occurrences.*

Definition 5: The task of ASPNO is to find the maximum nonoverlapping set. In this set, any two approximate occurrences of P in S are nonoverlapping.

B. NETTREE

In this subsection, we introduce the concept of the Nettree data structure at first. Then an example is employed to illustrate that an instance of approximate strict pattern matching can be expressed by a Nettree. Finally, we propose the algorithm CreNetTree which creates a Nettree for the problem. Nettree [25] is a kind of data structure that is similar to a tree data structure. The characteristics of Nettree are shown in Definition 6.

Definition 6: Nettree has four characteristics.

(1) A Nettree may have n roots, where $n > 1$.

(2) Any node except the root may have more than one parent and all its parents must be at the same level.

(3) A Nettree may have many nodes with the same label. But these nodes are on different level. In order to describe a node effectively, node i on the j^{th} *level is denoted by* n_j^i.

TABLE 2. The occurrences under the different methods.

Method	Number of occurrences	Occurrences
Traditional pattern matching	2	4 and 6
Strict pattern matching under no special condition	3	$\langle 1,2,4\rangle$, $\langle 1,3,4\rangle$, and $\langle 4,5,6\rangle$
Strict pattern matching under the one-off condition	1	Anyone of $\langle 1,2,4\rangle$, $\langle 1,3,4\rangle$, or $\langle 4,5,6\rangle$
Strict pattern matching under the nonoverlapping condition	2	$\langle 1,2,4\rangle$ and $\langle 4,5,6\rangle$

TABLE 3. A comparison of related works.

Related work	Matching/Mining	Traditional/strict	Type of matching	Length constraints	Type of condition
Navarro et al. [15]	Matching	Traditional	Exact	No	Traditional
Bille et al. [35]	Matching	Traditional/ Stricta	Exact	No	Traditional / No special
Wu et al. [28]	Matching	Strict	Approximate	Yes	No special
Li et al. [21]	Mining	Strict	-	No	No special
He et al. [33]	Matching	Strict	Approximate	Yes	One-off
Lam et al. [31]	Mining	Strict	-	No	One-off
Wu et al. [25]	Matching	Strict	Exact	Yes	Nonoverlapping
Ding et al. [26]	Mining	Strict	-	Yes	Nonoverlapping
Wu et al. [27]	Mining	Strict	-	Yes	Nonoverlapping
This paper	Matching	Strict	Approximate	Yes	Nonoverlapping

(4) There is more than one path from n_j^i to its descendant or its ancestor.

Although our previous work [25] employed Nettree to handle the issue of the strict pattern matching under the nonoverlapping condition, it is the exact version. In this paper, to deal with the approximate version, a special concept of Nettree is addressed.

Definition 7: NRPDC (Number of Roots Paths with Distance Constraints). We can see that there is more than one path from n_j^i to its ancestor. So there is more than one path from n_j^i to a root. The number of paths from n_j^i to the roots with Hamming distance d is called the Number of Roots Paths with Distance Constraints (NRPDC) and denoted by NR (n_j^i, d). For a root n_1^i, if s_i is equal to p_1, that is, $s_i = p_1$, the Hamming distance between s_i and p_1 is 0, and then NR $(n_1^i, 0) = 1$ and the other NR (n_1^i, d) $(d > 0)$ are 0. Otherwise, if s_i is not equal to p_1, that is, $s_i \neq p_1$, and then NR $(n_1^i, 1) = 1$ and the other NR (n_1^i, d) $(d \neq 0)$ are 0.

Definition 8: In a Nettree, a path $\langle n_1^{i_1}, n_2^{i_2}, \cdots, n_m^{i_m}\rangle$ is called a root-leaf path and its corresponding occurrence is $\langle i_1, i_2, \cdots, i_m\rangle$. The max root is called the rightmost root. Similarly, we can have the rightmost leaf, the rightmost child, and the rightmost parent. The rightmost root-leaf path iterates to obtain the rightmost child from the rightmost root or to obtain the rightmost parent from the rightmost leaf.

To illustrate the above concepts, a Nettree is shown in Fig. 1.

Example 5: In Fig. 1, for instance, s_3 is the corresponding subsequence of node n_2^3 and is equal to p_2, so node n_2^3 is a white node while s_3 is not equal to p_3, and hence node n_3^3 is a grey node. There are two numbers at the top left of each node which are used to indicate NR $(n_j^i, 0)$ and NR $(n_j^i, 1)$, respectively. Node n_3^8 is the rightmost child of node n_2^6 and the rightmost parent of node n_4^9. Path $\langle n_1^1, n_2^3, n_3^4, n_4^5\rangle$ is a root-leaf path and its corresponding occurrence is $\langle 1, 3, 4, 5\rangle$. It is easy to see the Hamming distance of occurrence $\langle 1, 3, 4, 5\rangle$ is 1 since node n_3^4 is a grey node. Node n_1^6 is the rightmost root. Nevertheless, path $\langle n_1^5, n_2^6, n_3^8, n_4^9\rangle$ is the rightmost root-leaf path since there is no root-leaf path from node n_1^6. The corresponding occurrence of path $\langle n_1^5, n_2^6, n_3^8, n_4^9\rangle$ is $\langle 5, 6, 8, 9\rangle$. Suppose path $\langle n_1^5, n_2^6, n_3^8, n_4^9\rangle$ is deleted, path $\langle n_1^2, n_2^3, n_3^5, n_4^8\rangle$ is the rightmost root-leaf path and the Hamming distance of its corresponding occurrence is 2 since there are two grey nodes, n_1^2 and n_4^8, in the path.

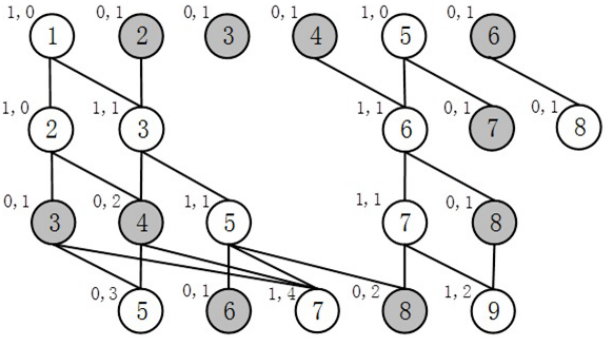

FIGURE 1. A Nettree with NRPDCs for each node. The two numbers at the top left of nodes are used to indicate $NR(n_j^i, 0)$ and $NR(n_j^i, 1)$. There are two kinds of nodes: grey and white. The grey node means that the corresponding subsequence is different from the corresponding subpattern, that is, it is an approximate matching, while the white node means that the corresponding subsequence is the same as the corresponding subpattern, that is, it is an exact matching.

Next, we will show how to update $NR(n_j^i, d)$ of a non-root node n_j^i.

Lemma 1: If $n_{j-1}^{r_q}$ and n_j^i satisfy gap constraint $p_{j-1}[min_{j-1}, max_{j-1}]p_j$, $n_{j-1}^{r_q}$ is a parent node of n_j^i. $N_R(n_j^i, d)$ can be updated as follows. If $s_i = p_j$, for all $d(0 \leq d \leq T)$, $N_R(n_j^i, d) += N_R(n_{j-1}^{r_q}, d)$. Otherwise, for all $d(1 \leq d \leq T)$, $N_R(n_j^i, d) += N_R(n_{j-1}^{r_q}, d-1)$ and $N_R(n_j^i, 0) = 0$.

Proof: If s_i is equal to p_j, that is, $s_i = p_j$, then after adding node n_j^i, the distance between $p_1 \cdots p_{j-1}p_j$ and

$s_{l_1} \ldots s_{l_{j-1}} s_i$ is the same as the distance between $p_1 \ldots p_{j-1}$ and $s_{l_1} \ldots s_{l_{j-1}}$, that is, $D(p_1 \ldots p_{j-1} p_j, s_{l_1} \ldots s_{l_{j-1}} s_i) = D(p_1 \ldots p_{j-1}, s_{l_1} \ldots s_{l_{j-1}})$. So $N_R(n_j^i, d)$ should add $N_R(n_{j-1}^{r_q}, d)$ if $n_{j-1}^{r_q}$ is a parent node of n_j^i. If s_i is not equal to p_j, that is, $s_i \neq p_j$, then after adding node n_j^i, the distance between $p_1 \ldots p_{j-}p_j$ and $s_{l_1} \ldots s_{l_{j-1}} s_i$ increases by 1, that is, $D(p_1 \ldots p_{j-1} p_j, s_{l_1} \ldots s_{l_{j-1}} s_i) = D(p_1 \ldots p_{j-1}, s_{l_1} \ldots s_{l_{j-1}}) + 1$. So $N_R(n_j^i, d)$ should add $N_R(n_{j-1}^{r_q}, d-1)$ when $d > 0$ and $N_R(n_j^i, d)$ is 0 when $d = 0$. ■

Lemma 2: If $\sum_{d=1}^{T} N_R(n_j^i, d) = 0$, n_j^i can be deleted or does not need to be created.

Proof: We can see that $D(p_1 \ldots p_{j-1} p_j, s_{l_1} \ldots s_{l_{j-1}} s_i)$ is no less than $D(p_1 \ldots p_{j-1}, s_{l_1} \ldots s_{l_{j-1}})$. Therefore, the Hamming distance is monotonous. If $\sum_{d=1}^{T} N_R(n_j^i, d) = 0$, this means that the distance of all the paths from node n_j^i to any root is greater than T. Therefore, the distance of all the paths from node n_k^i passing through node n_j^i to any root is greater than T since the Hamming distance is monotonous. Hence, node n_j^i can be deleted or does not need to be created. ■

Lemma 3: Suppose $n_{j-1}^{r_q}$ is a parent node of n_j^i according to the gap constraint. If $s_i \neq p_j$ and for all $d (1 \leq d \leq T-1)$, $\sum_{d=1}^{T-1} N_R(n_{j-1}^{r_q}, d) = 0$, the parent-child relationship between $n_{j-1}^{r_q}$ and n_j^i can be deleted.

Proof: According to Lemma 2, we know that $\sum_{d=1}^{T} N_R(n_{j-1}^{r_q}, d)$ is greater than 0, otherwise node $n_{j-1}^{r_q}$ is deleted. Because $\sum_{d=1}^{T-1} N_R(n_{j-1}^{r_q}, d) = 0$, $N_R(n_{j-1}^{r_q}, T)$ is greater than 0, which means that the distance of all paths from node $n_{j-1}^{r_q}$ to its roots is T. Since s_i is not equal to p_j, the distance of the paths from node n_j^i to its roots via node $n_{j-1}^{r_q}$ is $T+1$, which does not meet the criteria. Although $n_{j-1}^{r_q}$ is a parent node of n_j^i according to the gap constraint, the relationship fails to satisfy the criteria. Hence, the parent-child relationship between $n_{j-1}^{r_q}$ and n_j^i can be deleted. ■

Next, an example is used to illustrate the process that transforms an instance into a Nettree.

Example 6: Suppose we have sequence $S = s_1 s_2 s_3 s_4 s_5 s_6 s_7 s_8 s_9$ = AGGTAGAGA, pattern $P = p_1[min_1, max_1] p_2[min_2, max_2] p_3[min_3, max_3] p_4 = A[0,1]G[0,1]A[0,2]A$, and threshold $T = 1$.

We obtain the first letter s_1 in sequence S first. Since $s_1 = p_1$ = 'A', we create a root n_1^1 on the Nettree and $N_R(n_1^1, 0)$ and $N_R(n_1^1, 1)$ are 1 and 0, respectively. Then we obtain the second letter s_2 in the sequence. Now, we can see that the Nettree has only a root on the first level. So s_2 can be created on the first and second levels, respectively. Since s_2 ='G' $\neq p_1$, according to Definition 7, we create a root n_1^2 and $N_R(n_1^2, 0)$ and $N_R(n_1^2, 1)$ are 0 and 1, respectively. Since 2 - 1 - 1 = 0 which satisfies the gap constraints [0, 1], we create a node n_2^2 on the second level. Since s_2 ='G'= p_2, according to Lemma 1, we know that $N_R(n_2^2, 0)$ and $N_R(n_2^2, 1)$ are 1 and 0, respectively. Now, we deal with s_3. Since there is a node on the second level, s_3 can be created on the first, second, and third levels, respectively. Since s_3='G' $\neq p_1$, we create a root n_1^3 and $N_R(n_1^3, 0)$ and $N_R(n_1^3, 1)$ are 0 and 1, respectively. It is easy to see that both n_1^1 and n_1^2 can be parents according to the gap constraints [0, 1]. So $N_R(n_2^3, 0)$ and $N_R(n_2^3, 1)$ are both 1 since s_3 ='G'= p_2. Similarly, we create node n_2^3 and get $N_R(n_2^3, 0) = 0$ and $N_R(n_2^3, 1) = 1$. Now, we process s_4 = 'T'. It is easy to see that s_4 should be compared with p_1, p_2, p_3, and p_4, respectively. For the sake of conciseness, here we only talk about n_2^4. Since s_4='T' $\neq p_2$ ='G' and n_2^4 has two parents, n_1^2 and n_1^3, according to the gap constraints [0,1], we get $N_R(n_2^4, 0) = N_R(n_2^4, 1) = 0$. According to Lemma 3, n_2^4 is deleted. Hence, the Nettree is created and shown in Fig. 1.

From this example, we can see that the Nettree can be created by one-way scanning of the sequence and the benefits of NRPDC are threefold. (1) Some useless parent-child relationships can be deleted. n_1^6 can be a parent of n_2^7 according to the gap constraint A[0,1]G. But we can see that $N_R(n_1^6, 0) = 0$ and $s_7 \neq p_2$. According to Lemma 3, in the figure, the parent-child relationship between n_1^6 and n_2^7 is deleted. Similarly, the parent-child relationships between n_2^7 and n_3^8 and between n_3^4 and n_4^6 are deleted. (2) Some useless nodes can be deleted according to Theorem 2. For instance, in Fig. 1, nodes n_2^4, n_2^5, and n_3^6 are deleted. 3) There are $N_R(n_j^i, d)$ paths from node n_j^i to its roots with distance d. For instance, as we know that $N_R(n_3^5, 0)$ is 1, we can safely say that there is a path from a root to n_3^5 with distance 0 which is $\langle 1, 3, 5 \rangle$. Similarly, $N_R(n_3^5, 1)$ is also 1, so there is a path from a root to n_3^5 with distance 1, which is $\langle 2, 3, 5 \rangle$.

Algorithm CreNetTree therefore creates a Nettree and is shown as follows.

Algorithm 1 CreNetTree

Input: sequence S, pattern P, similarity constraint T
Output: *NetTree*

```
for i = 1 to n step 1 do;
    Create node n_1^i and calculate its NRPDCs according to Definition 7
    for j = 2 to min(m, i) step 1 do;
        Create node n_j^i;
        Update NRPDCs of node n_j^i according to Lemma 1;
        if sum_{d=1}^{T} N_R(n_j^i, d) = 0 then delete node n_j^i according to Lemma 2;
    end for
end for
```

C. NETASPNO ALGORITHM

In this subsection, we first show two lemmas. An illustrative example is used to show the principle of our algorithm. Finally, we propose the algorithm NETASPNO.

Lemma 4: Let A and B be two root-leaf paths without using the same node on the Nettree. The corresponding occurrences of A and B are two nonoverlapping occurrences.

Proof: Suppose two paths from root to leaf A and B are $\langle n_1^{a_1}, n_2^{a_2}, \ldots, n_m^{a_m}\rangle$ and $\langle n_1^{b_1}, n_2^{b_2}, \cdots, n_m^{b_m}\rangle$, respectively. We can safely say that, for any $i(1 \le i \le m)$, a_i is not equal to b_i, that is, $a_i \ne b_i$, since according to the definition of Nettree, nodes with the same label are on the different levels and A and B do not use the same nodes on the Nettree. Hence, $\langle a_1, a_2, \cdots, a_m\rangle$ and $\langle b_1, b_2, \cdots, b_m\rangle$ are two nonoverlapping occurrences. ■

Lemma 5: We can get a path from the rightmost leaf to its rightmost root without using the backtracking strategy.

Proof: Suppose the rightmost leaf is n_m^k. We can safely say that $\sum_{d=1}^{T} N_R(n_m^{a_m}, d)$ is greater than 0, that is, $\sum_{d=1}^{T} N_R(n_m^{a_m}, d) > 0$, otherwise, n_m^k should be deleted according to Lemma 2. Suppose $N_R(n_j^i, d) = k$ is greater than 0, which means that there are k paths from node n_j^i to its roots with distance d and if $N_R(n_j^i, d)$ is 0, there is no path from n_j^i to its roots with distance d. In the process of searching a path from node n_j^i to its root with distance d, if $s_{r_q} = p_{j-1}$, we can carry out iteration to select parent $n_{j-1}^{r_q}$ whose $N_R(n_{j-1}^{r_q}, d)$ is greater than 0, otherwise, we carry out iteration to select parent $n_{j-1}^{r_q}$ whose $N_R(n_{j-1}^{r_q}, d-1)$ is greater than 0. Iterating this process, we can get a path from n_m^k to its rightmost root without using the backtracking strategy. ■

The benefit of $N_R(n_j^i, d)$ therefore lies in the fact that the backtracking strategy can be avoided in a root-leaf path searching.

As we know, to solve the exact version of ASPNO, NETLAP-Best [25] finds the rightmost occurrence from the rightmost leaf and removes the found occurrence and other useless nodes. So in Example 3, if we use the similar principle of NETLAP-Best, we get the rightmost occurrence $\langle 5, 6, 8, 9\rangle$ from the rightmost leaf n_4^9 first. Then we find the next nonoverlapping occurrence. We can see that path $\langle n_1^2, n_2^3, n_3^5, n_4^8\rangle$ has two grey nodes and its corresponding occurrence is $\langle 2, 3, 5, 8\rangle$. So the Hamming distance between $s_2s_3s_5s_8$ and pattern P is 2, which is greater than 1. Therefore, we cannot select path $\langle n_1^2, n_2^3, n_3^5, n_4^8\rangle$ and have to get the rightmost occurrence $\langle 1, 3, 5, 8\rangle$ with distance 1 from the rightmost leaf n_4^8. There is no occurrence after removing $\langle 1, 3, 5, 8\rangle$. Therefore, there are two nonoverlapping occurrences, $\langle 5, 6, 8, 9\rangle$ and $\langle 1, 3, 5, 8\rangle$, using the similar principle of NETLAP-Best. However, it is easy to see that there are three nonoverlapping occurrences $\langle 5, 6, 8, 9\rangle$, $\langle 2, 3, 5, 7\rangle$, and $\langle 1, 2, 3, 5\rangle$, for this instance. Hence, we cannot employ the similar principle of NETLAP-Best [25] since it is easy to lose a feasible solution. To handle ASPNO, we propose the algorithm named NETASPNO, which obtains the rightmost root first. If there are some paths from the rightmost root to its m^{th} level leaves, then NETASPNO finds the rightmost root-leaf path from its rightmost leaf to obtain its corresponding occurrence. Otherwise, if there is no path from the rightmost root to its m^{th} level leaf, NETASPNO selects the next rightmost root. After obtaining the occurrence, NETASPNO deletes all the nodes of the rightmost root-leaf path. This process is iterated, until there are no occurrences that can be found. Example 7 is employed to illustrate the principle of NETASPNO directly.

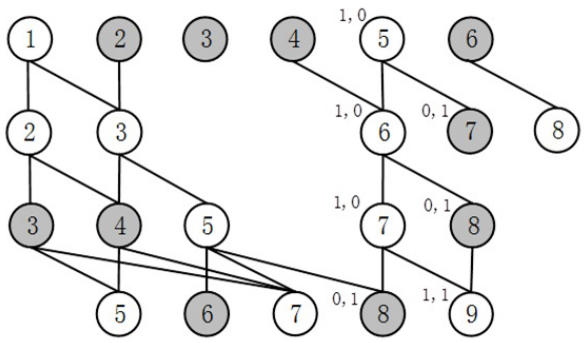

FIGURE 2. The Nettree and the subnettree with root n_1^5 and its NRPDCs for each node which are composed by two numbers, $NR(n_j^i, 0)$ and $NR(n_j^i, 1)$.

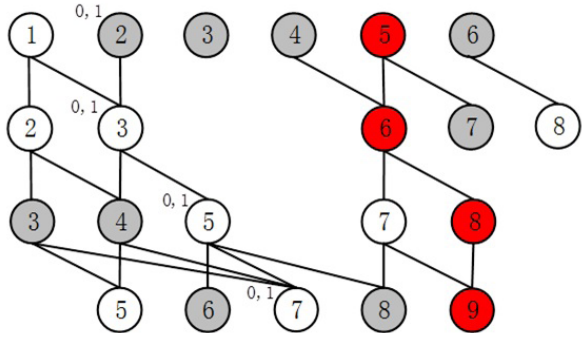

FIGURE 3. The subnettree with root n_1^2 and its NRPDCs and the Nettree after deleting occurrence $\langle 5, 6, 8, 9\rangle$. The red nodes are deleted.

Example 7: In this example, we also select the same pattern and sequence as in Example 6.

From Fig. 1, we can see that root n_1^6 is the rightmost root at first. However, there is no path from root n_1^6 to the fourth-level leaf. So NETASPNO finds the next rightmost root n_1^5. It is easy to see that there are some paths from n_1^5 to its fourth-level leaves, n_4^8 and n_4^9. Apparently, n_4^9 is the rightmost leaf of n_1^5. To avoid the efforts from other roots, such as root n_1^4, NETASPNO recalculates the NRPDCs of each node on the subnettree with root n_1^5 and the results are shown in Fig. 2. NETASPNO can select n_3^8 as the rightmost parent of leaf n_4^9 although $s_8 \ne p_3$, since $N_R(n_3^8, 1)$ is 1 which means that there is a path from root n_1^5 to n_3^8 with distance 1. $\langle 5, 6, 8, 9\rangle$ is the rightmost path from root n_1^5 to leaf n_4^9 via node n_3^8.

Now, NETASPNO deletes occurrence $\langle 5, 6, 8, 9\rangle$. Then NETASPNO selects root n_1^4. It is easy to see that root n_1^4 does not have a root-leaf path after deleting nodes n_1^5, n_2^6, n_3^8, and n_4^9, and neither does root n_1^3. The subnettree with root n_1^2 and its NRPDCs can be seen in Fig. 3. When NETASPNO selects root n_1^2, NETASPNO can find occurrence $\langle 2, 3, 5, 7\rangle$.

Finally, NETASPNO deletes occurrence $\langle 2, 3, 5, 7\rangle$ on the Nettree and recalculates NRPDCs for some nodes with

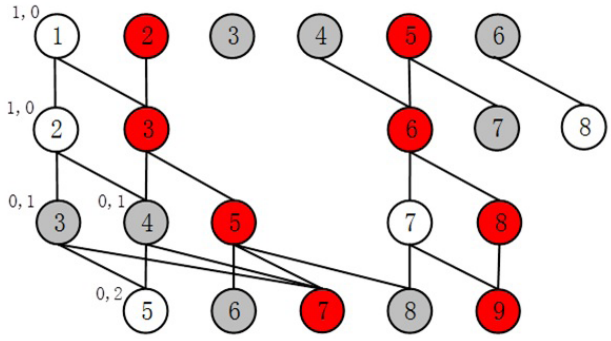

FIGURE 4. The subnettree with root n_1^1 and its NRPDCs and the Nettree after deleting occurrences ⟨2, 3, 5, 7⟩ and ⟨5, 6, 8, 9⟩.

root n_1^1. *The new Nettree can be seen in Fig. 4. It is easy to find the last occurrence* ⟨1, 2, 4, 5⟩ *on the new Nettree. Therefore NETASPNO finds three nonoverlapping occurrences,* ⟨1, 2, 4, 5⟩, ⟨2, 3, 5, 7⟩, *and* ⟨5, 6, 8, 9⟩ *for pattern P in sequence S with Hamming distance 1.*

Now, NETASPNO is shown as follows.

In Algorithm 2, NETASPNO adopts algorithm reachleaf to determine whether the r^{th} root can reach an m^{th} leaf or not. If the return value of algorithm reachleaf is -1, it means that there is no path from the r^{th} root to an m^{th} leaf, otherwise the function returns its rightmost leaf. The algorithm reachleaf is shown as follows.

Algorithm 2 NETASPNO

Input: sequence S, pattern P, and similarity constraint T
Output: nonoverlapping set C

```
Use Algorithm 1 to create NetTree
for r = the number of roots of NetTree downto 1 step -1 do;
    lf = reachleaf (r, NetTree, T)
    if lf > 0 then
        oc = getocc(lf , NetTree, T);
        C = C U oc
        NetTree− = oc; //delete oc on NetTree;
    end if
end for
```

In Algorithm 2, NETASPNO adopts the algorithm getocc to obtain the root-leaf path from leaf lf and its corresponding nonoverlapping occurrence. The algorithm getocc is shown as follows.

D. THE SPACE AND TIME COMPLEXITIES

Theorem 1: The space complexity of NETASPNO is $O(n * m * (T + g))$ *in the worst case, where n, m, g, and T are the length of sequence, the length of pattern, the maximal gap, and the similarity constraint, respectively.*

Proof: We can see that the space complexity of Nettree is $O(n*m*(T+g))$ in the worst case. The reasons are shown as follows. The Nettree has m levels, there are no more than n nodes on the Nettree in the worst case, and each node has no more than g parents in the worst case and stores $T+1$ values for NRPDCs. Therefore, the space and time complexities of creating the Nettree are $O(n * m * (T + g))$ and $O(n * m * T * g)$, respectively. There are no more than n nonoverlapping occurrences and each occurrence is composed of m indexes. So the space complexity of the nonoverlapping set C is $O(n * m)$. NETASPNO employs a Nettree and a nonoverlapping set to calculate and store the occurrences, respectively. Hence, the space complexity of NETASPNO is $O(n * m*(T + g))$. ■

Theorem 2: The time complexity of NETASPNO is $O(n * m^2 * T * g)$ *in the worst case.*

Proof: We have shown that the time complexity of Algorithm 1, created a Nettree, is $O(n * m * T * g)$ in Theorem 1. We analyze the time complexity of Algorithm 3 first. We know that each node has no more than g children. There are no more than $g * (i - 1)$ children on the i^{th} level. So no more than $g*m*(m-1)$ nodes need to recalculate their NRPDCs. Each node carries out the calculation $T + 1$ times for its NRPDCs. Hence, the time complexity of Algorithm 3 is $O(g*T*m*m)$ in the worst case. It is easy to see that the time complexity of Algorithm 4 is $O(g*T*m)$. There are no more than n roots on the Nettree. Therefore, the time complexity of NETASPNO is $O(n*m*T*g+n*(g*T*m*m+g*T*m)) = O(n * m^2 * T * g)$ in the worst case. ■

Algorithm 3 reachleaf

Input: root r, *NetTree*, and T
Output: the position of the rightmost leaf

```
start = end = r;
for l = 1 to m - 1 step 1 do
    for j = start to end step 1 do
        n = NetTree[l][j];
        nc = the number of children of n
        for k = 1 to nc step 1 do
            c = NetTree[l + 1][k]
            Recalculate NRPDCs for child c with the r^th root according to Lemma 1;
        end for
    end for
    start = the position of the first child on the l + 1^th level;
    end = the position of the last child on the l+1^th level;
    if start < 0 then return -1;
end for
lf = end;
return lf
```

IV. EXPERIMENTAL RESULTS AND ANALYSIS

A. EXPERIMENTAL ENVIRONMENT AND DATA

To evaluate the performance, all experiments are conducted on a laptop with an Intel (R) Core i7-5500U, with a 2.40 GHZ CPU, 4.00 GB of RAM, and Windows 7

Algorithm 4 getocc

Input: rightmost leaf lf, $NetTree$, and similarity constraint T
Output: nonoverlapping occurrence oc

```
distance = 0;
oc[m] = ct = NetTree[m][lf]; // ct means current node.
if ct.match=false then distance++; // ct.match = false means the subsequence is different from the corresponding subpattern
for j = m downto 2 step -1 do
    np= the number of parents of ct
    for i = np downto 1 step -1 do
        pt = ct[j];
        pd = 0;
        if pt.match = false then pd = 1;
        if pt.used = false and distance+pd ≤ T and ∑_{d=distance}^{T} N_R(n_j^{pt}, d) > 0 then
            distance+ = pd;
            oc[m − 1] = ct = pt;
            break;
        end if
    end for
end for
return oc
```

SP1 operating system. We also propose other two algorithms, NETLAP-Appro and NETROL (NETtree from Root tO Leaf). NETLAP-Appro adopts the same principle as NETLAP-Best and it iterates to find the rightmost root-leaf-path from the rightmost leaf while NETROL iterates to find the leftmost root-leaf path from the leftmost root. As we know that the time complexity of NETLAP-Best is $O(m * m * n * g)$ for the exact pattern matching issue. It is easy to get that the time complexity of NETLAP-Appro is also $O(m * m * n * g * T)$. So is NETROL. This means that the three algorithms have the same time complexity. We develop NETASPNO, NETLAP-Appro, and NETROL by VC++ 6.0. All these algorithms can be downloaded from http://wuc.scse.hebut.edu.cn/nettree/netaspno. To evaluate the performance of NETASPNO impartially, we also use the patterns (shown in Table 4) and the DNA sequences (shown in Table 5) as benchmark patterns and sequences, which were employed to evaluate the performances of NETLAP-Best [25], SONG [28] and SBO [36]. These real DNA sequences can be downloaded from http://www.ncbi.nlm.nih.gov.

B. CORRECTNESS

As we know, when similarity constraint T is 0, the approximate version transforms into the exact version automatically. To show the correctness of NETASPNO, in this subsection, we therefore set T as 0. The numbers of occurrences are shown in Table 6. The same results can be obtained using NETLAP-Best to deal with the exact version. Therefore, we can safely say that NETASPNO is correct.

TABLE 4. Patterns.

Name	Pattern
$P1$	a[0,3]t[0,3]a[0,3]t[0,3]a[0,3]t[0,3]a[0,3]t[0,3]a[0,3]t[0,3]a
$P2$	g[1,5]t[0,6]a[2,7]g[3,9]t[2,5]a[4,9]g[1,8]t[2,9]a
$P3$	g[1,9]t[1,9]a[1,9]g[1,9]t[1,9]a[1,9]g[1,9]t[1,9]a[1,9]g[1,9]t
$P4$	g[1,5]t[0,6]a[2,7]g[3,9]t[2,5]a[4,9]g[1,8]t[2,9]a[1,9]g[1,9]t
$P5$	a[0,10]a[0,10]t[0,10]c[0,10]g[0,10]g
$P6$	a[0,5]t[0,7]c[0,9]g[0,11]g
$P7$	a[0,5]t[0,7]c[0,6]g[0,8]t[0,7]c[0,9]g
$P8$	a[5,6]c[4,7]g[3,8]t[2,8]a[1,7]c[0,9]g
$P9$	c[0,5]t[0,5]g[0,5]a[0,5]a

TABLE 5. The real DNA sequences.

Name	From	Locus	Length
$S1$	Segment 1	CY058563	2286
$S2$	Segment 2	CY058562	2299
$S3$	Segment 3	CY058561	2169
$S4$	Segment 4	CY058556	1720
$S5$	Segment 5	CY058559	1516
$S6$	Segment 6	CY058558	1418
$S7$	Segment 7	CY058557	982
$S8$	Segment 8	CY058560	844

TABLE 6. The numbers of occurrences of $T = 0$.

Sequence	P1	P2	P3	P4	P5	P6	P7	P8	P9
S1	33	126	203	113	270	228	138	95	163
S2	19	142	228	133	270	233	164	91	188
S3	20	130	221	124	272	235	158	71	181
S4	29	108	178	101	205	184	132	57	139
S5	26	91	138	85	179	155	107	59	120
S6	19	79	135	72	173	146	102	49	121
S7	10	64	102	60	135	112	84	42	84
S8	5	54	78	47	90	86	65	33	73

C. PERFORMANCE

For a practical application, it is meaningless to set T very high, especially for a short pattern. We know that the length of $P9$ is 5. Therefore, in this paper, we set T as 1 or 2. Fig. 5 and Fig. 6 report the numbers of occurrences of $T = 1$ and $T = 2$, respectively.

To show the results concisely, we sum up the occurrences of the same pattern in the eight sequences of $T = 1$ and $T = 2$ which are shown in Tables 7 and 8, respectively. Since NETROL, NETLAP-Appro, and NETASPNO are heuristic algorithms, we select the max result obtained by the three algorithms as the best result. The results that are close to the best results are shown in bold in the tables.

The approximation ratio $\rho =$ (the result)/(the best result) is used to show the results visually. The results of $T = 1$ and $T = 2$ are shown in Fig. 7 and Fig. 8, respectively.

According to Table 7 and Table 8 and Fig. 5 and Fig. 6, we can say that NETASPNO has better performance than the others. In this paper, nine patterns are selected. NETASPNO obtains the best results on six patterns for both $T = 1$ and $T = 2$. For instance, from Table 7, we can see that NETROL and NETLAP-Appro find 2336 and 2222 occurrences for $P4$ in eight sequences, respectively, when T is 2, while NETASPNO finds 2502. From Table 6, we also

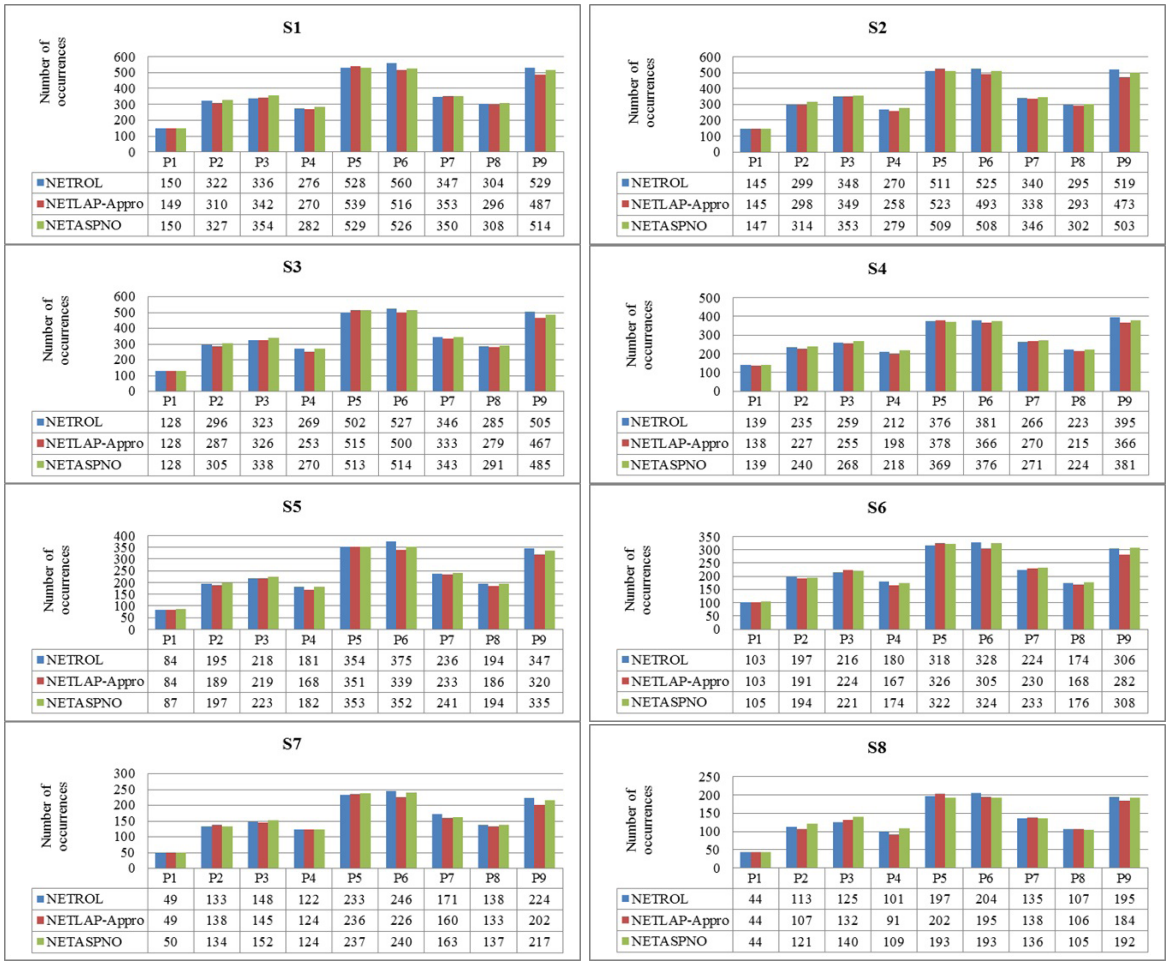

FIGURE 5. The number of occurrences of $T = 1$.

TABLE 7. When $T = 1$, the sum of the number of occurrences of pattern in the eight sequences.

Sequence	P1	P2	P3	P4	P5	P6	P7	P8	P9
NETROL	842	1790	1973	1611	3019	3146	2065	1720	3020
NETLAP-Appro	840	1747	1992	1529	3070	2940	2055	1676	2781
NETASPNO	850	1832	2049	1638	3025	3033	2083	1737	2935
The best result	850	1839	2052	1644	3074	3146	2099	1740	3022

TABLE 8. When $T = 2$, the sum of the number of occurrences of pattern in the eight sequences.

Sequence	P1	P2	P3	P4	P5	P6	P7	P8	P9
NETROL	1867	2785	2502	2336	4178	4716	3204	3076	5321
NETLAP-Appro	1869	2748	2524	2222	4626	4933	3171	3068	4952
NETASPNO	1952	2920	2760	2502	4385	4783	3297	3197	5305
The best result	1952	2920	2760	2502	4627	4933	3320	3197	5337

notice that the results of NETASPNO are the same with the best results on five patterns, $P1$, $P2$, $P3$, $P4$, and $P8$. This means that NETASPNO obtains the max results on all $5*8 = 40$ instances. Further, statistics show that NETASPNO obtains the max results 40 times and 49 times for $T = 1$ and $T = 2$, respectively. Therefore, NETASPNO is considerably better than the others. The reason for this is that NETASPNO employs a more effective strategy to find nonoverlapping occurrences.

We show the running time for $T = 1$ and $T = 2$ in Table 9 and Table 10, respectively.

From Table 9 and Table 10, we can see that NETASPNO is faster than the others in all instances. For instance, NETROL and NETLAP-Appro take 1.87 s and 1.30 s for $P1$,

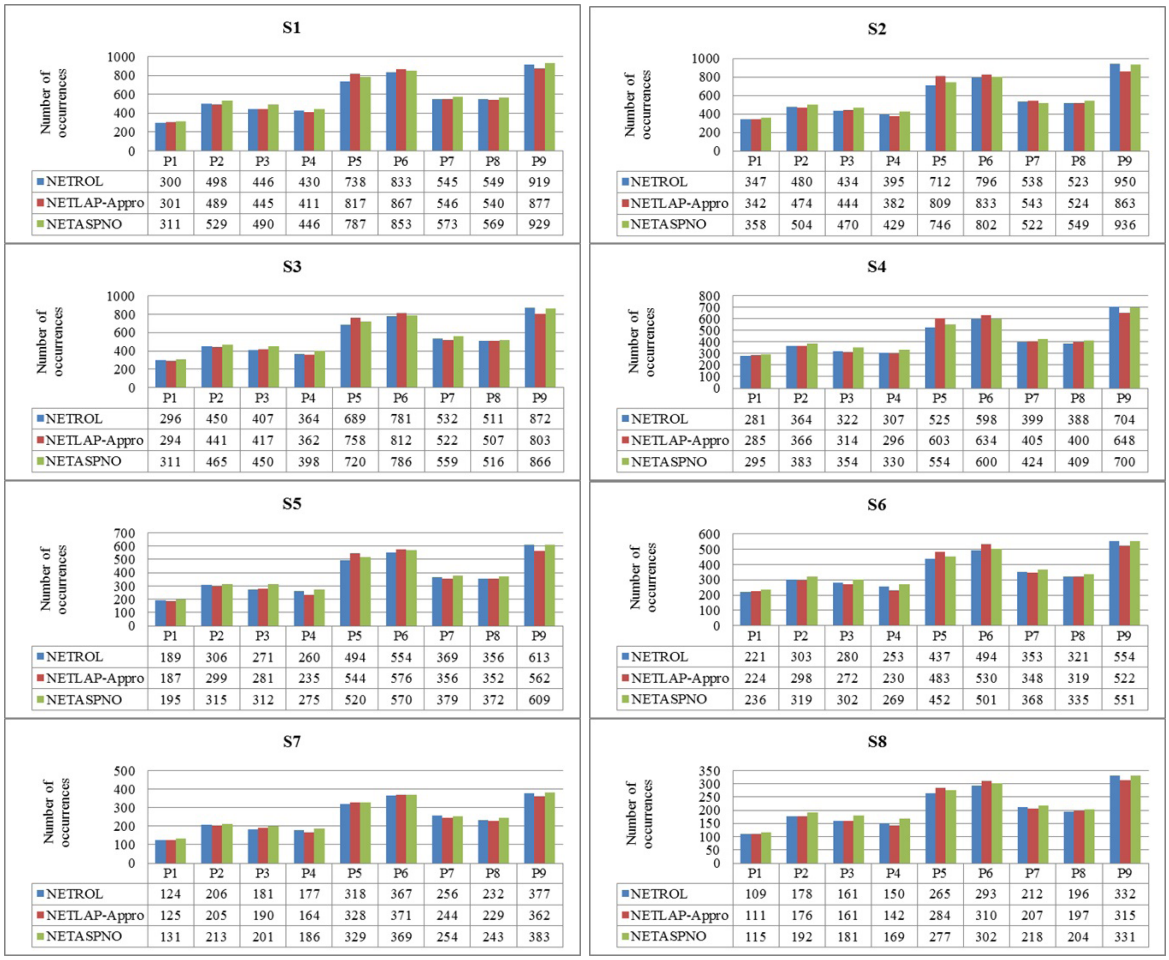

FIGURE 6. The number of occurrences of $T = 2$.

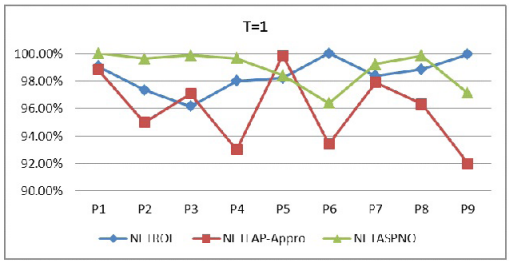

FIGURE 7. The approximation ratio of $T = 1$.

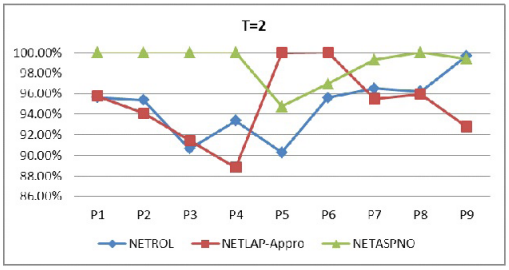

FIGURE 8. The approximation ratio of $T = 2$.

respectively while NETASPNO takes 0.77 s. Generally, NETASPNO is two to three times faster than the others. The reason is shown as follows. NETLAP-Appro adopts the same principle as NETLAP-Best which employs a more complex strategy to solve the issue. For example, in Example 3, NETLAP-Best gets the rightmost occurrence $\langle 5, 6, 8, 9\rangle$ from the rightmost leaf n_4^9 first. Then NETLAP-Best finds all useless nodes on the Nettree and deletes them after deleting nodes n_1^5, n_2^6, n_3^8, and n_4^9. Therefore, node n_3^7 must be found out and deleted. Apparently, NETASPNO does not need to find this kind of nodes. Hence, NETASPNO employs a more effective pruning strategy.

We can also see that the running time of NETROL and NETLAP-Appro is almost the same. For example, NETROL takes 76.04 s and NETLAP-Appro takes 77.67 s for all 72 instances in $T = 2$. The reason is that NETROL employs the similar principle as NETLAP-Appro.

From Table 9 and Table 10, we can see that the running time of NETASPNO for T=2 is almost twice that of $T = 1$. For example, the running time of NETASPNO for $P1$ with

TABLE 9. The running time for $T = 1$ (s).

Sequence	P1	P2	P3	P4	P5	P6	P7	P8	P9
NETROL	1.87	2.29	5.85	3.01	3.73	1.62	1.90	1.22	1.01
NETLAP-Appro	1.30	1.84	4.88	2.45	2.70	1.59	1.76	1.25	0.97
NETASPNO	0.77	1.08	2.03	1.36	1.23	0.76	0.98	0.63	0.61

TABLE 10. The running time for $T = 2$ (s).

Sequence	P1	P2	P3	P4	P5	P6	P7	P8	P9
NETROL	2.86	5.80	22.40	8.81	17.38	6.22	6.99	2.90	2.68
NETLAP-Appro	2.45	5.69	22.11	8.92	17.32	7.82	7.52	3.22	2.62
NETASPNO	1.48	1.98	3.73	2.56	2.20	1.52	1.90	1.37	1.09

$T = 2$ is 1.48 s while that for $T = 1$ is 0.77 s. All other experiments show the similar phenomenon. This phenomenon therefore verifies the correctness of time complexity of NETASPNO.

According to the above experimental results, we can safely say that NETASPNO has better performance than the other two algorithms.

V. CONCLUSION

In this paper, we address a type of approximate pattern matching, named approximate pattern matching under the nonoverlapping condition. Comparing with the exact pattern matching version, the new problem is more general and more challenging. We propose an effective algorithm, NETASPNO, which transforms an instance of approximate pattern matching into a Nettree at first. Due to the similarity constraint, some of the parent-child relationships cannot be selected to find a root-leaf path. A concept called NRPDCs is proposed to handle the issue. Then NETASPNO iterates to find the rightmost root-leaf path from the rightmost root as a nonoverlapping occurrence. It is not necessary for NETASPNO to detect the useless nodes, and therefore NETASPNO employs a more effective pruning strategy. Experimental results show that NETASPNO has better performance than the other competitive algorithms.

Nevertheless, the strategy of iteration to find the rightmost root-leaf path is a complete strategy under the exact pattern matching while this is a heuristic strategy under the approximate version. The reason for this is as follows. Under exact nonoverlapping pattern matching, we proved that if $\langle a, d\rangle$ and $\langle b, c\rangle$ ($a < b$ and $c < d$) are two suboccurrences of a subpattern [25], we can safely say that $\langle a, c\rangle$ and $\langle b, d\rangle$ are also two suboccurrences of a subpattern. To find the nonoverlapping occurrences, we can perform iterate to find the rightmost suboccurrence $\langle b, d\rangle$. But due to the similarity constraint, the conclusion cannot be eternally true under the approximate version. Here is an example. Suppose we have pattern $P = A[0, 1]B[0, 2]C[0, 1]D$ and sequence $S = s_1s_2s_3s_4s_5s_6s_7s_8 = AACBECDD$ under $T = 1$. It is easy to know that both suboccurrences $\langle 3, 6\rangle$ and $\langle 4, 5\rangle$ satisfy the subpattern $B[0, 2]C$ and the similarity constraint. But $\langle 3, 5\rangle$ does not satisfy the similarity constraint. We can find only one nonoverlapping occurrence $\langle 2, 4, 6, 8\rangle$ if we employ the strategy of the rightmost root-leaf path. Actually, there are two nonoverlapping occurrence, $\langle 1, 3, 6, 8\rangle$ and $\langle 2, 4, 5, 7\rangle$, for this instance. Therefore, a better performance algorithm should be studied in the future. Next, we will apply this method to mine approximate sequence patterns to find more valuable patterns in the sequences.

REFERENCES

[1] F. Claude, G. Navarro, H. Peltola, L. Salmela, and J. Tarhio, "String matching with alphabet sampling," *J. Discrete Algorithms*, vol. 11, pp. 37–50, 2012.
[2] G. Navarro, "Spaces, trees, and colors: The algorithmic landscape of document retrieval on sequences," *ACM Comput. Surv.*, vol. 46, no. 4, 2014, Art. no. 52.
[3] H. Le and V. K. Prasanna, "A memory-efficient and modular approach for large-scale string pattern matching," *IEEE Trans. Comput.*, vol. 62, no. 5, pp. 844–857, May 2013.
[4] H. Hu, K. Zheng, X. Wang, and A. Zhou, "GFilter: A general gram filter for string similarity search," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 4, pp. 1005–1018, Apr. 2015.
[5] F. Li, B. Yao, M. Tang, and M. Hadjieleftheriou, "Spatial approximate string search," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 6, pp. 1394–1409, Jun. 2013.
[6] P. Bille, J. Fischer, I. Li Gørtz, T. Kopelowitz, B. Sach, and H. W. Vildhøj, "Sparse text indexing in small space," *ACM Trans. Algorithms*, vol. 12, no. 3, p. 39, Jun. 2016.
[7] J. Liu, Z. M. Ma, and X. Feng, "Answering ordered tree pattern queries over fuzzy XML data," *Knowl. Inf. Syst.*, vol. 43, no. 2, pp. 473–495, 2015.
[8] G. Navarro and Y. Nekrich, "Time-optimal top-k document retrieval," *SIAM J. Comput.*, vol. 46, no. 1, pp. 80–113, 2015.
[9] Y.-X. Wu, L.-L. Wang, J.-D. Ren, W. Ding, and X.-D. Wu, "Mining sequential patterns with periodic wildcard gaps," *Appl. Intell.*, vol. 41, no. 1, pp. 99–116, 2014.
[10] K.-H. Chen, G. Huang, and R. C.-T. Lee, "Bit-parallel algorithms for exact circular string matching," *Comput. J.*, vol. 57, no. 5, pp. 731–743, May 2014.
[11] J. Lin, Y. Jiang, and D. Adjeroh, "Circular pattern discovery," *Comput. J.*, vol. 58, no. 5, pp. 1061–1073, May 2015.
[12] M. Crochemore, C. Iliopoulos, C. Makris, W. Rytter, A. Tsakalidis, and K. Tsichlas, "Approximate string matching with gaps," *Nordic J. Comput.*, vol. 9, no. 1, pp. 54–65, 2002.
[13] K. Nip, Z. Wang, and W. Xing, "A study on several combination problems of classic shop scheduling and shortest path," *Theor. Comput. Sci.*, vol. 22, pp. 175–187, Nov. 2016.
[14] K. Fredriksson and S. Grabowski, "Efficient algorithms for pattern matching with general gaps, character classes, and transposition invariance," *Inf. Retr.*, vol. 11, no. 4, pp. 335–357, Aug. 2008.
[15] G. Navarro and M. Raffinot, "Fast and simple character classes and bounded gaps pattern matching, with applications to protein searching," *J. Comput. Biol.*, vol. 10, no. 6, pp. 903–923, 2003.
[16] M. D. Retwitzer, M. Polishchuk, E. Churkin, I. Kifer, Z. Yakhini, and D. Barash, "RNAPattMatch: A Web server for RNA sequence/structure motif detection based on pattern matching with flexible gaps," *Nucl. Acids Res.*, vol. 43, no. W1, pp. W507–W512, 2015.
[17] C.-D. Tan, F. Min, M. Wang, H.-R. Zhang, and Z.-H. Zhang, "Discovering patterns with weak-wildcard gaps," *IEEE Access*, vol. 4, pp. 4922–4932, 2016.
[18] S. Ghosh, M. Feng, H. Nguyen, and J. Li, "Risk prediction for acute hypotensive patients by using gap constrained sequential contrast patterns," in *Proc. Annu. Symp. Amer. Med. Inform. Assoc. (AMIA)*, 2014, pp. 1748–1757.

[19] S.-J. Yen and Y.-S. Lee, "Mining non-redundant time-gap sequential patterns," *Appl. Intell.*, vol. 39, no. 4, pp. 727–738, Dec. 2013.
[20] X. Wang, L. Duan, G. Dong, Z. Yu, and C. Tang, "Efficient mining of density-aware distinguishing sequential patterns with gap constraints," in *Database Systems for Advanced Applications*. Bali, Indonesia: Springer, 2014, pp. 372–387.
[21] C. Li, Q. Yang, J. Wang, and M. Li, "Efficient mining of gap-constrained subsequences and its various applications," *ACM Trans. Knowl. Discovery Data*, vol. 6, no. 1, 2012, Art. no. 2.
[22] Y. Wu, S. Fu, H. Jiang, and X. Wu, "Strict approximate pattern matching with general gaps," *Appl. Intell.*, vol. 42, no. 3, pp. 566–580, Apr. 2015.
[23] H. Yang, L. Duan, B. Hu, S. Deng, W. Wang, and P. Qin, "Mining top-*k* distinguishing sequential patterns with gap constraint," *J. Softw.*, vol. 26, no. 11, pp. 2994–3009, 2015.
[24] H. F. Wang, L. Duan, J. Zuo, W. Wang, Z. Li, and C. Tang, "Efficient mining of distinguishing sequential patterns without a predefined gap constraint," *Chin. J. Comput.*, vol. 39, no. 10, pp. 1979–1991, 2016.
[25] Y. Wu, C. Shen, H. Jiang, and X. Wu, "Strict pattern matching under non-overlapping condition," *Sci. China Inf. Sci.*, vol. 60, no. 1, pp. 012101-1–012101-16, 2017.
[26] B. Ding, D. Lo, J. Han, and S.-C. Khoo, "Efficient mining of closed repetitive gapped subsequences from a sequence database," in *Proc. IEEE 25th Int. Conf. Data Eng. (ICDE)*, Shanghai, China, Mar./Apr. 2009, pp. 1024–1035.
[27] Y. Wu, Y. Tong, X. Zhu, and X. Wu, "NOSEP: Non-overlapping sequence pattern mining with gap constraints," *IEEE Trans. Cybern.*, to be published, doi: 10.1109/TCYB.2017.2750691.
[28] Y. Wu, Z. Tang, H. Jiang, and X. Wu, "Approximate pattern matching with gap constraints," *J. Inf. Sci.*, vol. 42, no. 5, pp. 639–658, 2016.
[29] H. Hu, H. Wang, J. Li, and H. Gao, "An efficient pruning strategy for approximate string matching over suffix tree," *Knowl. Inf. Syst.*, vol. 49, no. 1, pp. 121–141, 2016.
[30] X. Wu, X. Zhu, Y. He, and A. N. Arslan, "PMBC: Pattern mining from biological sequences with wildcard constraints," *Comput. Biol. Med.*, vol. 43, no. 5, pp. 481–492, Jun. 2013.
[31] H. Lam, F. Mörchen, D. Fradkin, and T. Calders, "Mining compressing sequential patterns," *Statist. Anal. Data Mining*, vol. 7, no. 1, pp. 34–52, 2012.
[32] X. Wu, J.-P. Qiang, and F. Xie, "Pattern matching with flexible wildcards," *J. Comput. Sci. Technol.*, vol. 29, no. 5, pp. 740–750, 2014.
[33] D. He, X. Wu, and X. Zhu, "SAIL-APPROX: An efficient on-line algorithm for approximate pattern matching with wildcards and length constraints," in *Proc. IEEE Int. Conf. Bioinform. Biomed. (BIBM)*, Nov. 2007, pp. 151–158.
[34] M. K. Warmuth and D. Haussler, "On the complexity of iterated shuffle," *J. Comput. Syst. Sci.*, vol. 28, no. 3, pp. 345–358, Jun. 1984.
[35] P. Bille, I. Li Gørtz, H. W. Vildhøj, and D. K. Wind, "String matching with variable length gaps," *Theor. Comput. Sci.*, vol. 443, pp. 25–34, Jul. 2012.
[36] Y.-X. Wu, F. Min, X.-D. Wu, and H. Jiang, "A heuristic algorithm for MPMGOOC," *Chin. J. Comput.*, vol. 34, no. 8, pp. 1452–1462, 2011.

YOUXI WU (M'17) received the Ph.D. degree in theory and new technology of electrical engineering from the Hebei University of Technology, Tianjin, China. He is currently a Ph.D. Supervisor and a Professor with the Hebei University of Technology. His current research interests include data mining and machine learning. He is a Senior Member of CCF.

SHASHA LI received the master's degree in computer science and technology from the Hebei University of Technology, Tianjin, China. Her current research interests include data mining.

JINGYU LIU received the Ph.D. degree in computer science and technology from the Beijing Institute of Technology, Beijing, China. He is an Associate Professor with the Hebei University of Technology, Tianjin. His current research interests include bioinformatics and data mining.

LEI GUO received the Ph.D. degree in theory and new technology of electrical engineering from the Hebei University of Technology, Tianjin, China. He is currently a Ph.D. Supervisor and a Professor with the Hebei University of Technology. His current research interests include image processing and machine learning.

XINDONG WU (F'11) received the Ph.D. degree from The University of Edinburgh, Edinburgh, U.K. He is a Yangtze River Scholar with the Hefei University of Technology, China, and a Professor of computer science with the School of Computing and Informatics, University of Louisiana at Lafayette. His research interests include data mining, big data analytics, knowledge based systems, and Web information exploration. He is the Steering Committee Chair of the IEEE International Conference on Data Mining and the Editor-in-Chief of the *Knowledge and Information Systems*. He is a fellow of the AAAS.