Noname manuscript No.
(will be inserted by the editor)

NetDAP: $(\delta, \gamma)-$Approximate pattern matching with length constraints

Youxi Wu · Jinquan Fan · Yan Li · Lei Guo · Xindong Wu

Received: date / Accepted: date

Abstract Pattern matching(PM) with gap constraints has been applied to compute the support of a pattern in a sequence, which is an essential task of the repetitive sequential pattern mining (or sequence pattern mining). Compared with exact PM, approximate PM allows data noise (differences) between the pattern and the matched subsequence. Therefore, more valuable patterns can be found. Approximate PM with gap constraints mainly adopts the Hamming distance to measure the approximation degree which only reflects the number of different characters between two sequences, but ignores the distance between different characters. Hence, this paper addresses (δ, γ) approximate PM with length constraints which employs local-global constraints to improve the accuracy of the PM, namely, the maximal distance between two corresponding characters is less or equal to the local threshold δ, and the sum of all the δ distances is also less or equal to the global threshold γ. To tackle the problem effectively, this paper proposes an effective online algorithm, named NetDAP, which employs a special designed data structure

Youxi Wu
School of Artificial Intelligence, Hebei University of Technology, Tianjin 300401, China
State Key Laboratory of Reliability and Intelligence of Electrical Equipment, Hebei University of Technology, Tianjin 300401, China
Hebei Key Laboratory of Big Data Computing, Tianjin 300401, China
E-mail: wuc567@163.com

Jinquan Fan
School of Artificial Intelligence, Hebei University of Technology, Tianjin 300401, China
E-mail: 18315623690@163.com

Yan Li
School of Economics and Management, Hebei University of Technology, Tianjin 300401, China
E-mail: lywuc@163.com

Lei Guo
State Key Laboratory of Reliability and Intelligence of Electrical Equipment, Hebei University of Technology, Tianjin 300401, China
E-mail: guoshengrui@163.com

Xindong Wu
Research Institute of Big Knowledge, Hefei University of Technology, Hefei 230009, China
Mininglamp Software Systems, Beijing 100084, China
E-mail: xwu@hfut.edu.cn

named approximate single-leaf Nettree. An approximate single-leaf Nettree can be created by adopting dynamic programming to determine the range of rootleaf, the minimal root, the maximal root, the range of nodes for each level, and the range of parents for each node. To improve the performance, two pruning strategies are proposed to prune the nodes and the parent-child relationships which do not satisfy the δ and γ distance constraints, respectively. Finally, extensive experimental results on real protein data sets and time series verify the performance of the proposed algorithm.

Keywords approximate pattern matching · gap constraints · wildcard · occurrence · (δ, γ) distance

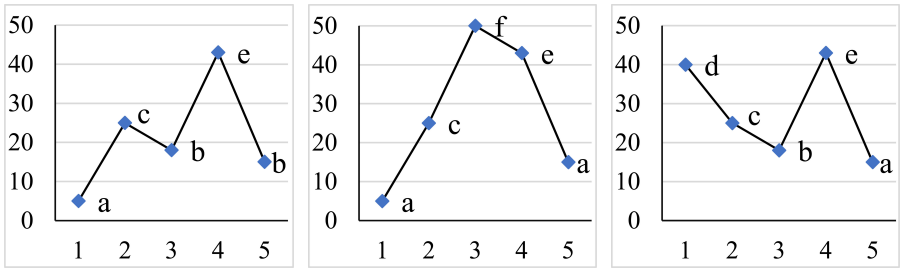
1 Introduction

Pattern matching (PM) (or string matching [1]) plays not only a very important role in many fields such as information retrieval [2, 3, 4], string similarity search detection [5], compressed data search [6], network application [7, 8], large-scale string matching [9], and patient detection [10], but is also a key issue of data mining [11, 12] and big wisdom [13]. To improve the flexibility of PM, researchers apply wildcards to PM. Therefore, users can control their demand more flexible. Hence, PM with wildcards has a significant impact on many fields, including error retrieval [14], pattern mining [15, 16, 17], pattern queries on XML data [18], and textual aggregation [19].

Compared with the traditional wildcard "?" or "*", PM with gap constraints is more flexible, which has attracted wide attention in recent years [20, 21] such as biological sequence research [22], customer purchase information mining [24], and feature selection of sequence classification [25]. The gap constraint is the amount of characters that can be wildcarded between two characters (or events), and it can be expressed as "$a[min, max]b$", where 'a' and 'b' are two characters and min and max represent the minimal and maximal don't care characters (wildcards), respectively [26]. In PM, a pattern with gap constraints can be defined as $p_1[min_1, max_1]p_2 \cdots [min_j, max_j]p_{j+1} \cdots [min_{m-1}, max_{m-1}]p_m$ [27, 28]. For example, $P =$ "$a[1, 2]c[0, 1]$" is a pattern with gap constraints, and its sub-pattern $a[1, 2]c$ indicates that there can be one or two don't care characters between 'a' and 'c', i.e. two traditional pattern substrings $a?c$ and $a??c$. PM with gap constraints can fulfill user inquiries more flexibly [29] and has two types: loose PM [30] and strict PM [31]. In traditional loose PM [32], an occurrence is represented by the tail position of a pattern in a sequence. However, in strict PM [33], an occurrence is represented by a group of position indexes. Obviously, the traditional loose PM ignores the matching process, while the strict PM preserves the details.

PM can be divided into exact PM and approximate PM. Exact PM requires that the pattern and the corresponding subsequence are the same, but limits the flexibility of the matching. For example, when dealing with biological protein sequences, noise will affect the accuracy of exact PM. Therefore, approximate PM [34, 35] can further reduce the influence of noise. The traditional approximate PM is mostly performed with Hamming distance [36]. However, there are some disadvantages when dealing with time series. For example, Figs. 1(a)(f), as time series, can be symbolized as "acbeb", "acfeb", "dcbeb", "bcbec", "acaec", and

"acbdc", respectively. "acfeb" and "dcbeb" can be approximate matched with "acbeb" with Hamming distance of 1. According to Figs. 1(b) and (c), we know that the trends are obviously different from that of Fig. 1(a). The reason is that Hamming distance does not consider the distance between two characters. However, according to Figs. 1(d), (e), and (f), the trends are similar to that of Fig. 1(a), although there are two characters that are different from "acbeb". The reason lies that the distance between two characters is 1. Hence, "bcbec", "acaec", and "acbdc" can be regarded as the approximate PM with the local constraint of 1 and the global constraint of 2. From this example, we can know that the approximate PM with local and global constraints has better performance than the traditional approximate PM with Hamming distance.



(a) acbeb (b) acfea (c) dcbea

(d) bcbec (e) acaec (f) acbdc

Fig. 1: Time series and their symbolized sequences

Regarding the issue above, this paper focuses on (δ, γ) approximate PM. The contributions of this paper are as follows:

- This paper addresses (δ, γ) approximate PM with length constraints, where the maximal distance between two characters is not greater than the local threshold δ, and the sum of all δ distances is also not greater than the global threshold γ.
- By employing an approximate single-leaf Nettree with (δ, γ) distance, we propose an effective online algorithm NetDAP (approximate single-leaf Nettree for Delta and gamma strict Approximate Pattern matching with length constraints) to find all occurrences and analyze the time and space complexities of NetDAP.

– Extensive experimental results not only show that NetDAP is better than other competitive algorithms, but also prove that (δ, γ) approximate PM with length constraints has better performance than the approximate PM with Hamming distance.

The rest of this paper is organized as follows. Section 2 briefly reviews the related work. We give the definition of (δ, γ) approximate PM with length constraints in Section 3. We introduce the approximate single-leaf Nettree, propose the NetDAP algorithm, and analyze the time and space complexities of NetDAP in Section 4. We report the performance of NetDAP and the (δ, γ) approximate PM in Section 5. We draw the conclusion in Section 6.

2 Related Work

PM with gap constraints is also called as string matching with gap constraints [37], where the gap constraints are wildcards with the lower and upper bounds [38, 39]. Thus, this kind of PM is more flexible and challenging than traditional PM with "?" and "*". PM with gap constraints is an essential issue in computer science. This kind of research involves many kinds of data structures and data types, such as multidimensional array [23], bipartite graph [40], suffix tree [41], Nettree [42], uncertain sequence [43], and time series [44]. More importantly, PM with gap constraints has been applied in many fields, such as music information retrieval [45], bioinformatics [46], and data mining [47]. For example, pattern "RK[2,3]DE[2,3]Y" can be used to find PROSITE protein site [46]. PM with gap constraints can also be applied in RNA sequence/structure motif detection [22]. Another important application area is data mining, since the purpose of PM with gap constraints is to calculate the number of occurrences (or support) for a pattern in a sequence which is also a key issue in repetitive sequential pattern mining (or sequence pattern mining) [48, 49]. Thus, various repetitive sequential pattern mining methods based on PM with gap constraints have been investigated, such as frequent pattern mining [50], tri-partition pattern mining [51], distinguishing pattern mining [42], and high utility pattern mining [52, 53].

From the perspective of occurrence, PM with gap constraints can be divided into two types, loosing PM and strict PM [29]. To represent an occurrence, the former uses the tail position of a pattern in a sequence, while the latter employs a group of positions of each subpattern in a sequence. In general, the studies in music information retrieval [45] and bioinformatics [22, 46] employ loosing PM, while the studies in data mining [38, 39, 50] adopt strict PM. Strict PM can be further subdivided into three types, no-condition [26, 27], one-off condition [54, 55], and nonoverlapping condition [33, 50, 56]. Example 1 illustrates the strict PM under the three conditions.

Example 1 Suppose we have S=ATATTA and pattern P=A[0,1]T[0,1]A. All occurrences for pattern P in sequence S are shown in Fig. 2.

We use a set of position indexes to indicate an occurrence. Under no-condition, each character can be used more than once. Therefore, there are three occurrences: $< 1, 2, 3 >$, $< 3, 4, 6 >$, and $< 3, 5, 6 >$. The nonoverlapping condition means that each subsequence can be used to match different subpatterns. Although s_3 is used in $< 1, 2, 3 >$ and $< 3, 4, 6 >$, it matches p_3 and p_1, respectively. Hence, $< 1, 2, 3 >$

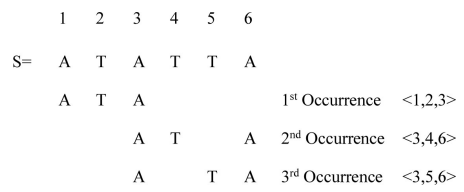


Fig. 2: All occurrences for P in S

Table 1: Comparison of related work

Related work	Gap constraint	Matching /Mining	Type of condition	Type of matching	Length constraints
Fischer et al. [57]	No	Matching	Loosing PM	Exact	No
Manber et al. [58]	Yes	Matching	Loosing PM	Exact	No
Li et al. [25]	Yes	Mining	No condition	Exact	Yes
Xie et al. [54]	Yes	Mining	One-off	Exact	Yes
Wu et al. [50]	Yes	Mining	Nonoverlapping	Exact	Yes
Min et al. [59]	Yes	Matching	No condition	Exact	Yes
Arslan et al. [40]	Yes	Matching	One-off	Exact	Yes
Shi et al. [31]	Yes	Matching	Nonoverlapping	Exact	Yes
Wu et al. [20]	Yes	Matching	No condition	Hamming distance	Yes
This paper	Yes	Matching	No condition	(δ, γ)distance	Yes

and $< 3, 4, 6 >$ are two nonoverlapping occurrences. Under the one-off condition, pattern P only has one occurrence, since each character in the given sequence can be used no more than once. For example, if $< 1, 2, 3 >$ is selected, $< 3, 4, 6 >$ is an illegal occurrence, since s_3 is used twice.

From the above example, we can know that if the no-condition result is deemed to be a complete set, the results of the nonoverlapping condition and the one-off condition will belong to different subsets. Nettree data structure has been employed to tackle these three conditions clearly and effectively. However, the computational complexities of the three conditions are significantly different. For example, once the Nettree is established, the number of occurrences for the no-condition problem can be calculated synchronously [60]. Our previous work proved that the nonoverlapping condition problem is in P and an iterative searching strategy was applied to find all the nonoverlapping occurrences in the Nettree [33, 50]. The one-off condition problem is NP-hard since it can be reduced from the iterated shuffle problem [61]. Thus, a heuristic strategy is employed to find the one-off occurrences [55]. Therefore, the no-condition problem is easier to be solved and is the key issue problem of other two problems.

Since PM with gap constraints has been applied to compute the support in repetitive sequential pattern mining, the related pattern mining studies are shown in Table 1.

From Table 1, we know that References [20, 59] are the most similar studies. Reference [59] addressed an exact PM. Reference [20] focused on an approximate PM with Hamming distance and proposed an effective algorithm, named SONG, which employed a special designed data structure named single-leaf Nettree to tackle the length and similarity constraints. SONG has to scan more characters to find all occurrences with the same first position. Hence, SONG is not an online algorithm. However, this paper focuses on an approximate PM with (δ, γ) dis-

tance and propose an effective online algorithm, named NetDAP, which is a more challenging and practical issue in time series applications.

3 Problem Definition

Definition 1 A pattern with gap constraints can be expressed as $P = p_1[min_1, max_1]p_2 \dots [min_j, max_j]p_{j+1} \dots [min_{m-1}, max_{m-1}]p_m$, where $1 \leq j \leq m$, $p_j \in \Sigma$, min_j and max_j as two non-negative integers, are the minimal and maximal gap constraints, respectively, and Σ is a collection of all events. Sequence S with length n can be written as $s_1 s_2 \dots s_i \dots s_n$, where $1 \leq i \leq n$ and $s_i \in \Sigma$.

Definition 2 Let patterns $P = p_1 p_2 \dots p_m$ and $Q = q_1 q_2 \dots q_m$. δ-distance (local constraint) refers to the absolute distance between two characters p_i and q_i, denoted by $D_\delta(p_i, q_i) = |p_i - q_i|$, $1 \leq i \leq m$. γ-distance (global constraint) is the sum of the δ-distances, and $D_\gamma(P, Q) = \sum_{i=1}^{m} |p_i - q_i|$.

Example 2 Suppose $P = p_1 p_2 p_3$=abc and $Q = q_1 q_2 q_3$=acd. According to Definition 2, $D_{\delta 1} = |p_1 - q_1| = 0, D_{\delta 2} = |p_2 - q_2| = 1, D_{\delta 3} = |p_3 - q_3| = 1$, and $D\gamma(P, Q)$=0+1+1=2.

Definition 3 Given a threshold δ and γ, a set of m integers $I =< i_1, i_2 \dots i_m >$ is a (δ, γ) approximate occurrence of pattern P in sequence S, if and only if:

$$min_{j-1} \leq i_j - i_{j-1} - 1 \leq max_{j-1} \tag{1}$$

$$D_\delta(s_{i_1}, p_1) \leq \delta, D_\delta(s_{i_2}, p_2) \leq \delta, \dots, D_\delta(s_{i_m}, p_m) \leq \delta \tag{2}$$

$$D_\gamma(s_{i_1} s_{i_2} \dots s_{i_m}, p_1 p_2 \dots p_m) \leq \gamma \tag{3}$$

$$0 < MinLen < i_m - i_1 + 1 \leq MaxLen \tag{4}$$

,where $MinLen$ and $MaxLen$ represent the minimal and maximal length constraints, respectively. Gap constraints can determine the occurrence span. For example, the lower and upper bounds of an occurrence are $m + \sum_{k=1}^{m-1} min_k$ and $m + \sum_{k=1}^{m-1} max_k$, respectively.

Definition 4 $N(S, P, \delta, \gamma)$ is the number of all (δ, γ) approximate occurrences.

Our problem is to calculate the number of (δ, γ) approximate occurrences for pattern P in sequence S, i.e. $N(S, P, \delta, \gamma)$.

Example 3 Given pattern $P = p_1[min_1, max_1]p_2[min_2, max_2]p_3$=a[0,1]b[0,1]a, sequence $S = s_1 s_2 s_3 s_4 s_5$= abcab, local threshold δ=1, global threshold γ=2 and length constraint $MinLen$=3, $MaxLen$=5.

According to subsequence a[0,1]b, the number of wildcards between 'a' and 'b' is 0, 1. We can know that $< 1, 2, 4 >$ is an exact occurrence, since $p_1 = s_1$=a, $p_2 = s_2$=b, $p_3 = s_4$=a, $min_1 \leq 2 - 1 - 2 \leq max_1$, and $min_2 \leq 4 - 2 - 1 \leq max_2$. But $< 1, 3, 4 >$ is not an exact occurrence, since p_2 is not equal to s_3. However, $< 1, 3, 4 >$ is an approximate matching which satisfies the δ constraint, since $|p_2 - s_3| = |b - c| = 1 \leq \delta$. The spans of $< 1, 2, 4 >$ and $< 1, 3, 4 >$ are 4-1+1=4 which satisfy the length constraints. $< 1, 2, 3 >$ is an approximate occurrence

satisfying Hamming distance of 1, but it is not an approximate matching that satisfies the δ constraint, since $|p_3 - s_3| = |c - a| = 2$ is greater than δ. Hence, there are four occurrences in this example: $< 1, 2, 4 >, < 1, 3, 4 >, < 1, 3, 5 >$, and $< 2, 3, 4 >$.

4 Proposed Algorithm

To solve our problem effectively, in this paper, we specially design a data structure, named single-leaf Nettree. Using this structure, an online algorithm, named NetDAP, is proposed. Finally, we analyze the time and space complexities of NetDAP.

4.1 Nettree

Nettree [50], as an extension of the tree structure, may have more than one root and more nodes with the same name. A node, denoted by n_j^i, may have more than one parent.

Definition 5 In a Nettree, a leaf on the m-th level is called an absolute leaf.

Definition 6 In a Nettree, a full path is a path from a root to an absolute leaf.

Lemma 1 *Each occurrence of pattern P in sequence S can be represented by a full path which corresponds to an occurrence.*

Proof Our previous work [60] has proved that all occurrences for pattern P in sequence S can be transformed into a Nettree, and each full path corresponds to one occurrence in a Nettree. Hence, each occurrence can be represented by a full path in a Nettree, i.e. full path $< n_1^{i_1}, n_2^{i_2}, \ldots, n_m^{i_m} >$ corresponds to occurrence $< i_1, i_2, \ldots, i_m >$.

According to Definition 3, we know that an approximate occurrence should satisfy both length constraints (the minimal and maximal length constraints) and (δ, γ) approximate constraints. Although multi-dimensional array [51] can be employed to tackle our problem, it is a sparse array. Thus, the array-based algorithm is less efficient. Nettree data structure [50] can also be used here and can avoid the processing of sparse data, but the algorithm using nettree is also less efficient, since it is very complex and needs to deal with the length and approximate constraints at the same time. An effective method is to calculate the number of occurrences with the same i_1, which employs the single-root Nettree [20]. However, this method has to pre-scan some characters after s_{i_1} in advance. Thus, it is not an online method. To handle the problem effectively, in this paper, we propose an effective online algorithm which employs a special designed data structure, named single-leaf Nettree and can calculate the number of occurrences with the same i_m. Based on single-leaf Nettree, we propose two pruning strategies, which can further avoid calculating the useless nodes.

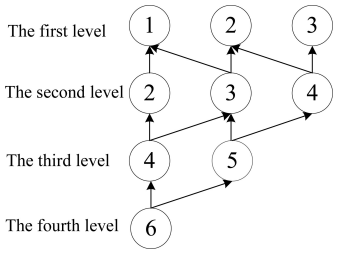


Fig. 3: A single-leaf Nettree. Node 6 in the fourth level is the only leaf in the Nettree. Thus, the Nettree is called single-leaf Nettree. Nodes 1, 2, and 3 in the first level are the roots. Node 3 in the second level is denoted by n_2^3 which has two parents: n_1^1 and n_1^2. Node n_2^3 has two children: n_3^4 and n_3^5. The arrow between two nodes represents parent-child relationship. The single-leaf Nettree is created according to the direction of the arrows.

4.2 Approximate single-leaf Nettree with (δ, γ) distance

Based on Nettree, we propose single-leaf Nettree at first. To create a single-leaf Nettree effectively, we iterately create the parents of current nodes. However, the range of parents depends on the range of nodes of this level which lies on the range of roots, since the length constraints should be considered. Therefore, the first step is to calculate the range of the roots. Then, the min and max brothers are designed. Furtherly, the min and max parents are proposed. To tackle the (δ, γ) distance effectively, we further propose the concept of approximate single-leaf Nettree. To reduce redundant computing, two pruning strategies are designed. Finally, the algorithm named LeafSolution is proposed, which can calculate the number of occurences whose last position is i_m.

Definition 7 A Nettree with only one leaf is called a single-leaf Nettree.

Fig. 3 shows a single-leaf Nettree. There is only one leaf n_4^6 and two nodes with the same ID: n_2^3 and n_1^3. Node n_3^4 has two parents: n_2^2 and n_2^3. There are two paths from leaf n_4^6 to roots, i.e. $< n_1^1, n_2^2, n_3^4, n_4^6 >$ and $< n_1^1, n_2^3, n_3^4, n_4^6 >$. Their corresponding occurrences are $< 1, 2, 4, 6 >$ and $< 1, 3, 4, 6 >$, respectively.

4.2.1 The min and max roots

Definition 8 In a single-leaf Nettree with leaf n_m^f, the min and max roots are denoted by $MinRt(n_m^f)$ and $MaxRt(n_m^f)$, respectively.

$MinRt(n_m^f)$ and $MaxRt(n_m^f)$ can be calculated according to Equations (5) and (6), respectively. The reason is shown as follows. We know that the upper bound of $MinRt(n_m^f)$ is $f - MaxLen + 1$ according to the length constraints and another upper bound of $MinRt(n_m^f)$ is $f - m + 1 - \sum_{k=1}^{m-1} max_k$ according to the gap constraints. $MinRt$ should select the smaller one from the above two values. Hence, $MinRt(n_m^f)$ can be calculated according to Equation (5). According to the length and gap constraints, the lower bounds of $MaxRt(n_m^f)$ are $f - MinLen + 1$ and $f - m + 1 - \sum_{k=1}^{m-1} min_k$, respectively. $MaxRt(n_m^f)$ should be no less than 1.

Therefore, $MaxRt(n_m^f)$ should select the larger one from the above three values. Hence, $MaxRt(n_m^f)$ can be calculated according to Equation (6).

$$MinRt(n_m^f) = max(f - MaxLen + 1, f - m + 1 - \sum_{k=1}^{m-1} max_k) \tag{5}$$

$$MaxRt(n_m^f) = min(f - MinLen + 1, f - m + 1 - \sum_{k=1}^{m-1} min_k, 1) \tag{6}$$

4.2.2 The min and max brothers

Now, we calculate the range of nodes for each level, i.e. the min and max brothers, based on the min and max roots.

Definition 9 In a single-leaf Nettree with root n_m^f, the min and max brothers on the j-th level are the minimal and maximal nodes, denoted by $MinBr(n_m^f, j)$ and $MaxBr(n_m^f, j)$, respectively.

$MinBr(n_m^f, j)$ and $MaxBr(n_m^f, j)$ can be calculated according to Equations (7) and (8), respectively. The reason is shown as follows. For p_j from the j-th level nodes in a single-leaf Nettree with leaf n_m^f, the maximal span of sub-pattern $p_j[min_j, max_j]p_{j+1} \dots [min_{m-1}, max_{m-1}]p_m$ is $m - j + \sum_{k=m-1}^{j} max_k$. Therefore, according to the gap constraints, the lower bound of $MinBr(n_m^f, j)$ is $f - m + j - \sum_{k=m-1}^{j} max_k$. According to $MinRt(n_m^f, j)$, the minimal span of $p_1[min_1, max_1]p_2 \dots [min_{j-1}, max_{j-1}]p_j$ is $j-1+\sum_{k=1}^{j-1} min_k$. Therefore, the minimal span from $MinRt(n_m^f)$ to the j-th level is $MinRt(n_m^f) + j - 1 + \sum_{k=1}^{j-1} min_k$. Hence, $MinBr(n_m^f, j)$ can be obtained according to Equation (7). Similarly, we can have Equation (8).

$$MinBr(n_m^f, j) = max(f - m + j - \sum_{k=m-1}^{j} max_k, MinRt(n_m^f) + j - 1 + \sum_{k=1}^{j-1} min_k) \tag{7}$$

$$MaxBr(n_m^f, j) = min(f - m + j - \sum_{k=m-1}^{j} min_k, MaxRt(n_m^f) + j - 1 + \sum_{k=1}^{j-1} max_k) \tag{8}$$

4.2.3 The min and max parents

At last, we determine the range of parents for each node based on the min and max brothers.

Definition 10 The min and max parents of node n_j^i are denoted by $MinPrt(n_j^i, j)$ and $MaxPrt(n_j^i, j)$, respectively.

$MinPrt(n_j^i, j)$ and $MaxPrt(n_j^i, j)(1 < j \leq m)$ can be calculated according to Equations (9) and (10), respectively. The reason is shown as follows. According to the gap constraints, the first parent of node n_j^i should be no less than $i - max_{j-1} - 1$ and $MinBr(n_m^f, j-1)$. Hence, $MinPrt(n_j^i, j)$ can be calculated according to Equation (9). Similarly, we can have Equation (10).

$$MinPrt(n_j^i, j) = max(i - max_{j-1} - 1, MinBr(n_m^f, j-1)) \tag{9}$$

$$MaxPrt(n_j^i, j) = min(i - min_{j-1} - 1, MaxBr(n_m^f, j-1)) \tag{10}$$

Now, a single-leaf Nettree can be created according to the min and max parents, the min and max brothers, and the min and max roots. An illstrative example is shown as follows.

Example 4 Given sequence $S = s_1s_2s_3s_4s_5s_6s_7$=BBBCEBA, pattern P=A[0,1] C[0,1]E[0,1]B, $MinLen$=4, and $MaxLen$=6. We create a single-leaf Nettree with leaf 6 as follows.

1) According to P and S, we know that $min_1 = min_2 = min_3$=0, $max_1 = max_2 = max_3$=1, m=4, n=10, and the leaf of the single-leaf Nettree is n_6^4.

2) We determine the range of roots. According to Equations (5) and (6), we know that $MinRt(n_4^6) = max(6-6+1, 6-4+1-3) = 1$ and $MaxRt(n_4^6) = min(6-4+1, 6-4+1-0, 1)$ =3. Hence, the single-leaf Nettree has three roots, i.e. n_1^1, n_1^2, and n_1^3.

3) We calculate the range of each level. For the 3rd level, according to Equations (7) and (8), we know that $MinBr(n_4^6, 3) = max(6-4+3-3, 1+3-1+0) = 3$ and $MaxBr(n_4^6, 3) = min(6-4+3-0, 3+3-1+3) = 5$. Hence, there are three nodes on the 3rd level, i.e. n_3^3, n_3^4, and n_3^5. Similarly, we can calculate MinBr and MaxBr for the 2nd level, i.e. $MinBr(n_4^6, 3) = max(6-4+2-3, 1+2-1+0) = 2$ and $MaxBr(n_4^6, 2) = min(6-4+2-0, 3+2-1+3) = 4$. Thus, there are three nodes on the 2nd level, i.e. n_2^2, n_2^3, and n_2^4.

4) We create the parents for each node. According to Equations (9) and (10), the min and max parents of node n_4^6 are $min(6-1-1, 4) = 4$ and $max(6-0-1, 5) = 5$, respectively. Hence, leaf n_4^6 has two parents: n_3^4 and n_3^5. Similarly, node n_3^4 has two parents: n_2^2 and n_2^3. Node n_3^5 has two parents: n_2^3 and n_2^4. Node n_2^2 has one parent: n_1^1. Node n_2^3 has two parents: n_1^1 and n_1^2. Node n_2^4 has two parents: n_1^2 and n_1^3. The single-leaf Nettree is shown in Fig.3.

4.2.4 Approximate single-leaf Nettree

Definition 11 The number of leaf paths which satisfies the γ-distance is called the number of γ-distance leaf paths, denoted by $NL(n_j^i, d_\gamma)$ $(0 \leq d_\gamma \leq \gamma)$.

For leaf n_m^i, if $D_\delta(s_i, p_i)$ is not greater than threshold δ, then $NL(n_j^i, d_\gamma)$ will be updated according to Equation(11):

$$NL\left(n_m^i, d_\gamma\right) = \begin{cases} 1 \; d_\gamma = D_\delta\left(s_i, p_m\right) \\ 0 \; d_\gamma \neq D_\delta\left(s_i, p_m\right) \end{cases} \tag{11}$$

When we set threshold γ=2, each node has three $NL(n_j^i, d_\gamma)$. For example, if $D_\delta(s_i, p_i) = 2$, then $NL\left(n_1^i, 2\right) = 1, NL\left(n_1^i, 0\right) = 0$, and $NL\left(n_1^i, 1\right) = 0$.

For node n_j^i, if nodes $n_{j+1}^{r_q}$ and n_j^i satisfy the gap constraints, $p_{j-1}[min_{j-1}, max_{j-1}]p_j$, $NL(n_j^i, d_\gamma)$ will be updated according to Equation (12). The reason is shown as follows. If $s_i = p_j$, then $D_\delta(s_i, p_j) = 0$. Therefore, when we add node n_j^i, and the distance between $p_j p_{j+1} \dots p_m$ and $s_i s_{l_{j+1}} \dots s_{l_m}$ is equal to the distance between $p_{j+1} \dots p_j$ and $s_{l_j+1} \dots s_{l_m}$, i.e. $D_\gamma(p_j p_{j+1} \dots p_m, s_i s_{l_{j+1}} \dots s_{l_m}) = D\gamma(p_{j+1} \dots p_m, s_{l_{j+1}} \dots s_{l_m})$. Hence, $NL\left(n_j^i, d_\gamma\right) += NL\left(n_{j+1}^{r_q}, d_\gamma\right)$.

If $s_i \neq p_j$, then $d_\delta = D_\delta\left(s_i, p_j\right)$. Therefore, we add node n_j^i, the distance between $p_j p_{j+1} \dots p_m$ and $s_i s_{l_{j+1}} \dots s_{l_m}$ will increase by d_δ, i.e. $D_\gamma(p_j p_{j+1} \dots p_m, s_i s_{l_{j+1}} \dots s_{l_m}) = D_\gamma\left(p_{j+1} \dots p_m, s_{l_{j+1}} \dots s_{l_m}\right) + d_\delta$. Hence, if $d_\delta \leq d_\gamma \leq \gamma$, then $NL\left(n_j^i, d_\gamma\right) += NL\left(n_{j-1}^{r_q}, d_\gamma - d_\delta\right)$, and if $0 \leq d_\gamma < d_\delta$, then $NL\left(n_j^i, d_\gamma\right) = 0$.

$$NL\left(n_j^i, d_\gamma\right) = \begin{cases} \sum_{q=m-1}^{t} NL\left(n_{j+1}^{r_q}, d_\gamma\right) & s_i = p_j, 0 \leq d_\gamma \leq \gamma \\ \sum_{q=m-1}^{t} NL\left(n_{j+1}^{r_q}, d_\gamma - d_\delta\right) & s_i \neq p_j, d_\delta \leq d_\gamma \leq \gamma \\ 0 & s_i \neq p_j, 0 \leq d_\gamma \leq d_\delta \end{cases} \tag{12}$$

, where node $n_{j+1}^{r_q}$ is a parent of n_j^i, t represents the number of parents of n_j^i, and $d_\delta = D_\delta(s_i, p_j)$.

4.2.5 Pruning strategies

According to Definition 3, if $D_\delta(s_i, p_j) > \delta$, node n_j^i can not be created. To further reduce redundant computing, we propose two pruning strategies, node pruning strategy and parent-child relationship pruning strategy.

- **Node pruning strategy:** If node n_j^i satisfies $\sum_{d_\gamma=0}^{\gamma} NL\left(n_j^i, d_\gamma\right) = 0$, then it will be pruned.
- **Parent-child relationship pruning strategy:** Suppose node $n_{j-1}^{r_q}$ is a parent of node n_j^i. If $s_i \neq p_j, d_\delta = D_\delta\left(s_i, p_j\right)$, and$\sum_{d_\gamma=0}^{\gamma-D_\delta} NL\left(n_{j-1}^{r_q}, d_\gamma\right) = 0\,(0 \leq d \leq \gamma - d_\delta)$, then the parent-child relationship between $n_{j-1}^{r_q}$ and n_j^i will be pruned.

The reasons are shown as follows.

Suppose $\sum_{d_\gamma=0}^{\gamma} NL\left(n_j^i, d_\gamma\right) = 0$. Hence, there is no path from n_j^i to its root whose γ-distance is less than or equal to γ. Therefore, we can safely delete node n_j^i.

If node $n_{j-1}^{r_q}$ exists, then $\sum_{d_\gamma=0}^{\gamma} NL\left(n_{j-1}^{r_q}, d_\gamma\right) \neq 0$. Since $\sum_{d_\gamma=0}^{\delta-d_\delta} NL\left(n_{j-1}^{r_q}, d_\gamma\right) = 0\,(0 \leq d \leq \gamma - d_\delta)$, $\sum_{d_\gamma=\gamma-d_\delta+1}^{\gamma} NL\left(n_{j-1}^{r_q}, d_\gamma\right) > 0$. Suppose there is a path from node $n_{j-1}^{r_q}$ to its leaf, whose γ-distance is no less than $\gamma - d_\delta + 1$. Since $s_i \neq p_j$ and $d_\delta = D_\delta(s_i, p_j)$, the γ-distance of the path from node n_j^i via its parent $n_{j-1}^{r_q}$ to its leaf will be greater than threshold γ. Therefore, there is no path from node n_j^i via its parent $n_{j-1}^{r_q}$ to its leaf whose γ-distance is less than γ. Hence, the parent-child relationship between $n_{j-1}^{r_q}$ and n_j^i can be pruned.

4.2.6 Leaf solution

Definition 12 The number of approximate occurrences with leaf n_m^f is called the leaf solution, denoted by $LS(n_m^f)$, which is the sum of the number leaf paths of its roots.

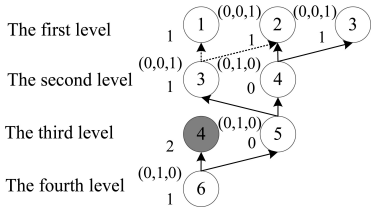


Fig. 4: The approximate single-leaf Nettree with of leaf n_4^6 with ($\delta = 1, \gamma = 2$). The gray nodes do not satisfy the approximate constraints and are pruned. The dashed-line arrows do not satisfy the parent-child relationships and are pruned. There are two types of numbers outside each node. One is on the top and the other is below. The below number indicates the δ- distance of current node, and the top numbers indicate the leaf paths with γ- distance (from 0 to γ). For example, the below number of leaf n_4^6 is 1, which means the δ- distance of node n_4^6, i.e. $|p_4 - s_6|$. The top numbers are (0,1,0), which are the leaf paths with γ-distances 0, 1, and 2, respectively. When the top numbers are all 0, i.e. (0,0,0), these information can be omitted, since the node is pruned, such as n_1^1 and n_3^4.

$LS(n_m^f)$ can be calculated according to Equation (13). The reason is shown as follows. Apparently, there are $NL(n_1^i, d_\gamma)$ full paths from leaf n_m^f to its root whose γ-distance are d_γ. According to Lemma 1, those full paths correspond to $NL(n_1^i, d_\gamma)$ occurrences whose γ-distance are d_γ. Therefore, there are $\sum_{d_\gamma=0}^{\gamma} NL\left(n_1^i, d_\gamma\right)$ occurrences from n_m^f to n_1^i. The range of leaves of n_m^f is from $MinRt(n_m^f)$ to $MaxRt(n_m^f)$. Hence, $LS(n_m^f)$ is $\sum_{i=MinRt(n_m^f)}^{MaxRt(n_m^f)} \sum_{d_\gamma=0}^{\gamma} NL\left(n_1^i, d_\gamma\right)$.

$$LS(n_m^f) = \sum_{i=MinRt(n_m^f)}^{MaxRt(n_m^f)} \sum_{d_\gamma=0}^{\gamma} NL\left(n_1^i, d_\gamma\right) \tag{13}$$

Example 5 We use the same pattern and sequence in Example 4 to illustrate the principle of the algorithm LeafSolution, which creates an approximate single-leaf Nettree with (δ, γ) distance and calculates its solution.

1) We know that $s_6 \neq p_4$, $D_\delta(s_6, p_4) = 1$. According to Definition 11, $NL(n_4^6, 0)$, $NL(n_4^6, 1)$, and $NL(n_4^6, 2)$ are 0, 1, and 0, respectively.

2) According to node prunning strategy, we can prune node n_4^3 since $D_\delta(s_3, p_4) = 2 > \delta$ and $\sum_{d_\gamma=0}^{\gamma} NL\left(n_3^4, d_\gamma\right) = 0$.

3) Since s_3='B' and p_2='C', $D_\delta(s_2, p_2) = 1$. The gap between 3 and 5 is 1. Therefore, s_3 satisfies threshold δ and the gap constraint. According to Equation (12), $NL(n_2^3, 0) = 0, NL(n_2^3, 1) = NL(n_3^5, 0) = 0$, and $NL(n_2^3, 2) = NL(n_3^5, 2) = 1$.

4) According parent-child relationship pruning strategy, we prune the parent-child relationships between n_2^3 and n_1^1, and between n_2^3 and n_1^2 since $D_\delta(s_1, p_1) = D_\delta(s_2, p_1) = 1 < \delta$ and $NL(n_2^3, 0), NL(n_2^3, 1)$, and $NL(n_2^3, 2)$ are 0, 0, and 1, respectively. Hence, $NL(n_1^2, 0), NL(n_1^2, 1)$, and $NL(n_1^2, 2)$ are 0, 0, and 1, respectively. Similarly, we create the approximate single-leaf Nettree shown in Fig. 4.

5)From Fig. 4, there are two roots: n_1^2 and n_1^3. Thus, $LS(n_4^6) = NL(n_1^2, d_\gamma) + NL(n_1^3, d_\gamma)$ =0+0+1+0+0+1=2.

Algorithm 1 LeafSolution

Input: Pattern P, Sequence S, approximate threshold δ, γ, leaf f
Output: $LS(n_m^f)$

```
Create leaf n_m^f and initialize NL(n_m^f, d_γ) according to Equation (11);
Calculate the min and max roots according to Equations (5) and (6), respectively;
for j = m − 1 to 1 step 1 do
    Calculate the min and max brothers according to Equations (7) and (8);
    for u = 1 to MaxBr(n_m^f, j) − MinBr(n_m^f, j) step 1 do
        k = get the u-th node on the j-1 th level;
        if D_δ(n_j^k, p_j) ≤ δ then
            Calculate the min and max parents of node n_j^i according to Equations (9) and (10);
            for i = MinPrt(n_j^k, j) to MaxPrt(n_j^k, j) step 1 do
                Update NL(n_j^i, d_γ) according to Equation (12);
                Prune invalid nodes and invalid parent-child relationships according to the pruning strategies;
            end for
        end if
    end for
end for
Calculate LS(n_m^f) according to Equation (13);
return LS(n_m^f);
```

The algorithm LeafSolution is shown as follows.

4.3 NetDAP

For a sequence with length n, the last leaf is n, but the first single-leaf Nettree is not 1, because some head characters do not satisfy either the gap constraints or the length constraints. Therefore, the range of leaves should be calculated.

Definition 13 The position of the first character which satisfies the length and gap constraints is called the minimal leaf, denoted by $MinLf$.

$MinLf$ can be calculated according to Equation (14). The reason is shown as follows. It is easy to know that the first leaf should be no less than $MinLen$. Hence, the lower bound of $MinLf$ is $Minlen$ according to the length constraints. However, according to the gap constraint, the lower bound of $MinLf$ is $m + \sum_{k=1}^{m-1} min_k$. We should select the larger one from the above two values. Hence, we can calculate $MinLf$ according to Equation (14).

$$MinLf = max(Minlen, m + \sum_{k=1}^{m-1} min_k) \tag{14}$$

Our problem is to calculate $N(S, P, \delta, \gamma)$ which is the sum of all leaf solutions, i.e. $N(S, P, \delta, \gamma)$ can be calculated according to Equation (15).

$$N(S, P, \delta, \gamma) = \sum_{f=MinLf}^{n} LS\left(n_m^f\right) \tag{15}$$

Algorithm 2 NetDAP

Input: Pattern P, Sequence S, approximate threshold δ, γ
Output: $N(S, P, \delta, \gamma)$
1: Calculate $MinLf$ according to Equation (14);
2: **for** $f = MinLf$ to n step 1 **do**
3: $N(S, P, \delta, \gamma)$ $+=LeafSolution(S, P, \delta, \gamma, f)$; //According to Equation (15)
4: **end for**
5: Return $N(S, P, \delta, \gamma)$;

We use Example 6 to show the principle of NetDAP.

Example 6 Given sequence $S = s_1s_2s_3s_4s_5s_6s_7$= BBBCEAB, pattern P=A[0,1] C[0,1]E[0,1]B, $MinLen$=4, $MaxLen$=6, δ=1, and γ=2.

According to Equation (14), we know that $MinLf = max(4, 4 + 0 + 0 + 0) = 4$. Thus, we select s_4 as the single-leaf Nettree with leaf n_4^4. According to gap constraints, node n_4^4 has only one parent, node n_3^3. Since s_3=B and p_3=E, we know that $D_\delta(s_3, p_3) = 3 > \delta$. Hence, we do not create the approximate single-leaf Nettree with leaf n_4^4. Similarly, since s_5=E and p_4=B, we know that $D_\delta(s_5, p_4) = 3 > \delta$. We do not create the approximate single-leaf Nettree with leaf n_4^5 either .

The approximate single-leaf Nettree with leaf n_4^6 is introduced in Example 5 shown in Fig. 4, and its leaf solution is 2. Since $D_\delta(s_7, p_4) = 0$, leaf n_4^7 can be created. According to the principle of Example 5, we can create the approximate single-leaf Nettree of leaf n_4^7 shown in Fig.5, and its leaf solution is $LS(n_4^7) = 0+0+1+0+1+1+0+0+1 = 4$. Hence, $N(S, P, \delta, \gamma) = LS(n_4^6)+LS(n_4^7) = 2+4 = 6$, i.e. there are six occurrences for pattern P in sequence S.

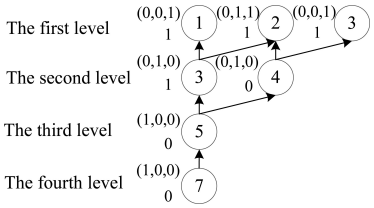


Fig. 5: The approximate single-leaf Nettree of leaf n_4^7

Obviously, NetDAP is an online algorithm which is shown as follows.

4.4 Time and space complexity analysis

The space complexity of NetDAP is $O(\gamma * g * m^2 + n)$, where γ, g, m, and n are the given global constraint threshold, the maximal gap, the pattern length, and the sequence length, respectively. The reason is shown as follows. The LeafSolution algorithm creates a single-leaf Nettree. In the Nettree, each node has no more than g parents. Thus, the $(m - j)$-th level has at most $j * g$ nodes $(1 \leq j < m)$. Therefore, a single-leaf Nettree has no more than $O(g * m^2)$ nodes. Each node

stores $\gamma + 1$ $NL(n_j^i, d)$ values. Hence, The space complexity of the LeafSolution algorithm is $O(g * m^2 * \gamma)$. Further more, the space complexity of NetDAP is $O(g * m^2 * \gamma + n)$ since the single-leaf Nettree does not need to be stored in the LeafSolution algorithm.

The time complexity of NetDAP is $O(\gamma * g^2 * m^2 * n)$. The reason is shown as follows. We analyze the time complexity of LeafSolution at first. Obviously, the time complexities of lines 10 and 11 are $O(\gamma)$. The loop count of line 9 is $O(g)$. The time complexity of line 8 is $O(1)$. The loop counts of lines 5 and 3 are $O(m*g)$ and $O(m)$, respectively. The time complexities of lines 2 and 4 are both $O(m)$. The time complexities of lines 1 and 16 are $O(\gamma)$ and $O(\gamma * m * g)$, respectively. Thus, the time complexity of LeafSolution is $O(\gamma*g^2*m^2)$. Since there are no more than n leaves, LeafSolution runs no more than n times. Hence, the time complexity of NetDAP is $O(\gamma * g^2 * m^2 * n)$.

5 Experimental Results and Analysis

Subsection 5.1 introduces the principle of the competitive algorithms. Subsection 5.2 shows the experimental environment and data sets. Subsection 5.3 reports the running time performance. Subsection 5.4 analyzes the influence of different approximate parameters on the results and running time. Subsection 5.5 compares the matching performance of the approximate PM with (δ, γ) distance and with Hamming distance.

5.1 Baseline methods

In this paper, we first address the problem of (δ, γ) approximate PM with length constraints and propose an algorithm named NetDAP. To verify the performance of NetDAP, we also propose five competitive algorithms, PAIG-DAP, NAM-APPRO, SONG-D, NetDAP-N, and NetDAP-P. Meanwhile, we also select two competitive algorithms, SONG and NETASPNO, to evaluate the matching performance. All these algorithms are summarized as follows.

- PAIG-DAP algorithm: PAIG-DAP is based on PAIG [59] which is the state-of-the-art algorithm and has been applied to mine frequent sequence patterns [62], frequent patterns with weak-wildcard gaps [23], and frequent tri-partition alphabets patterns [51]. We improve PAIG to (δ, γ) approximate pattern matching algorithm and compare it with NetDAP in this paper.
- NAM-APPRO algorithm: NAM-APPRO is based on NAMIC algorithm [60] and employs Nettree to solve (δ, γ) approximate PM problem. Nettree data structure has been applied to tackle various pattern matching [33, 35] and sequential pattern mining issues [38, 50]. We report the superiority of the data structure through comparative experiments.
- SONG-D algorithm: SONG-D is based on SONG [20], which is used to solve approximate PM problem with (δ, γ) distance.
- NetDAP-N and NetDAP-P algorithms: To verify the effectiveness of the pruning strategies, NetDAP-N and NetDAP-P algorithms employ node and parent-child relationship pruning strategies, respectively.

Table 2: Data sets

Name	UniProtKB	Gene	Length
$S1$	A0A3Q3L924	Mastacembelus armatus	4,064
$S2$	A0A3B4UA59	Seriola dumerili	4,295
$S3$	A0A2N8LHF0	Mycobacterium sp. ENV421	5,190
$S4$	A0A078K571	Plasmodium yoeli	5,316
$S5$	A0A383R7Y4	Paenibacillus alvei	6,312
$S6$	A0A1E7WGV1	Duganella sp. HH101	6,697
$S7$	A0A3B0KSX7	Drosophila guanche	7,426
$S8$	A0A388K3K1	Chara braunii	8,399
$S9$	SDB4	ASTRAL95_1_171	109,424
$S10$	UPI000A932905	PATRIC	204,482
$S11$	TS1	WormsTwoClass TEST 46	150
$S12$	TS2	WormsTwoClass TEST 44	150
$S13$	TS3	BirdChicken	150
$S14$	TS4	BeetleFly	150

– SONG and NETASPNO algorithms. To evaluate the approximate matching performance, SONG [20] and NETASPNO [35] are selected, which are approximate PM with Hamming distance under no condition and nonoverlapping condition, respectively.

All these algorithms and NetDAP can be downloaded at http://wuc.scse.hebut.edu.cn/ and are developed in VC++ 6.0. All experiments run on an Inter(R) Core(TM) i5-3230 CPU with 2.60GHz, 8.0GB of RAM, Windows 10 Professional, and 64-bit operating systems.

5.2 Experimental environment and data sets

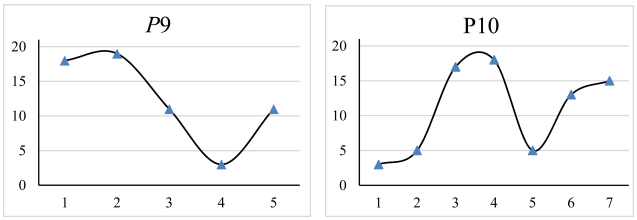
This paper uses the real protein sequences and time series shown in Table 2. The 8 protein sequences can be downloaded from https://www.uniprot.org/. Meanwhile, to verify the matching performance, we select the WormsTwoClass data (rows 44 and 46) in the UAR time series data set and other two data sets, BirdChicken and BeetleFly in the time series classification, which can be downloaded from http://www.as.uar.edu/eamonn/time-series-data/ and http://www.timeseriesclassification.com/, respectively.

We use the symbolization tool SAX [64] to symbolize the time series and replace the serialization result with 20 characters (A-T).

To explore the influence of different factors on the matching performance, we design ten patterns shown in Table 3. $P1 - P3$ are random patterns. To verify the influence of pattern length on the running time, we design patterns $P4 - P5$ which have the same gap constraints, but different pattern length. To study the influence of maximal gap, we add patterns $P6 - P8$ which have the same length constraint, but different pattern gap. To verify the matching performance of (δ, γ) distance in time series, we add patterns $P9 - P10$ and the corresponding trend figures are given in Fig. 6.

Table 3: Patterns

Name	Pattern	MinLen	MaxLen
$P1$	D[2,9]F[1,8]E[0,10]K[2,6]A[3,7]G	11	40
$P2$	E[1,9]D[0,8]A[2,7]L[1,8]Q[4,9]R[3,8]U	13	38
$P3$	S[1,9]K[3,11]D[1,7]F[1,9]P[3,8]G[2,6]A[0,5]L	15	55
$P4$	D[0,7]F[0,7]S[0,7]D[0,7]F[0,7]S	16	40
$P5$	D[0,7]F[0,7]S[0,7]D[0,7]F[0,7]S[0,7]D[0,7]F	16	40
$P6$	G[0,8]D[0,8]S[0,8]A[0,8]F[0,8]T	11	35
$P7$	G[0,7]D[0,7]S[0,7]A[0,7]F[0,7]T	11	35
$P8$	G[0,6]D[0,6]S[0,6]A[0,6]F[0,6]T	11	35
$P9$	R[0,5]S[0,5]K[0,5]C[0,5]K	10	24
$P10$	C[0,5]E[0,5]Q[0,5R[0,5]E[0,5]M[0,5]O	10	24



(a) P9 (b) P10

Fig. 6: Trends of patterns $P9$ and $P10$

Table 4: The number of occurrences on $S1 - S8$ with $(\delta = 1, \gamma = 2)$

Sequence	$P1$	$P2$	$P3$	$P4$	$P5$	$P6$	$P7$	$P8$	Total
$S1$	34	22	20	292	54	299	215	119	1055
$S2$	81	8	8	890	415	508	254	131	2295
$S3$	110	102	33	93	10	658	367	183	1556
$S4$	42	3	0	174	19	99	51	19	407
$S5$	208	114	50	397	181	534	266	136	1886
$S6$	66	182	116	172	98	804	399	186	2023
$S7$	250	44	38	384	231	668	385	249	2249
$S8$	237	35	112	618	264	1233	717	396	3612

5.3 Running time performance

Obviously, if the approximate thresholds are too large, any sequence will be matched with the pattern. Thus, it will lead to serious deviation of matching results, which has no practical significance. To report the performance of NetDAP, in this subsection, we show the running time and the number of occurrences for patterns $P1-P8$ in sequences $S1 - S8$ with parameters (δ= 1, γ=2) in detail in Fig. 7 and Table 4, respectively. To further demonstrate the influence of different approximate parameters on the results and the running time, parameters $(\delta = 2, \gamma = 2), (\delta = 1, \gamma = 3)$, $(\delta = 2, \gamma = 3)$, $(\delta = 1, \gamma = 4)$, and $(\delta = 2, \gamma = 4)$ are selected. The total running time and the number of occurrences for $P1 - P8$ in $S1 - S8$ are shown in Figs. 8, 9, 10, 11, 12, and Table 5, respectively.

S1	P1	P2	P3	P4	P5	P6	P7	P8
PAIG-DAP	157	203	265	172	297	187	172	156
NAM-APPRO	42.2	43.7	51.5	46.9	51.6	42.1	40.7	39.1
SONG-D	6.2	6.2	9.3	12.5	15.6	11.1	9.4	9.3
NetDAP-P	4.7	6.3	11.1	9.4	12.5	9.4	9.4	7.8
NetDAP-N	4.7	6.3	7.8	6.2	7.8	7.8	6.3	4.7
NetDAP	4.7	4.7	6.3	6.2	7.8	7.8	6.2	4.5

S2	P1	P2	P3	P4	P5	P6	P7	P8
PAIG-DAP	156	218	315	187	313	203	187	156
NAM-APPRO	46.9	50.1	54.7	51.6	59.4	50.2	46.9	46.9
SONG-D	7.8	7.8	15.6	10.9	17.2	14.1	11.1	9.4
NetDAP-P	6.2	6.4	10.9	9.4	15.6	14.1	9.4	7.8
NetDAP-N	4.7	6.5	7.8	7.9	9.4	7.8	6.3	6.2
NetDAP	4.6	6.3	7.8	7.8	9.4	6.3	6.2	6.1

S3	P1	P2	P3	P4	P5	P6	P7	P8
PAIG-DAP	204	281	344	219	344	234	219	188
NAM-APPRO	61.2	70.4	81.2	61.3	68.8	70.3	68.8	53.1
SONG-D	15.6	9.4	11.1	12.5	17.2	15.7	12.5	11.1
NetDAP-P	6.2	7.8	12.5	10.9	17.1	12.5	10.9	9.4
NetDAP-N	6.2	7.8	7.9	7.8	9.4	7.8	7.8	6.2
NetDAP	6.2	6.2	7.9	7.8	9.2	7.8	7.8	6.1

S4	P1	P2	P3	P4	P5	P6	P7	P8
PAIG-DAP	219	266	359	219	344	235	218	187
NAM-APPRO	62.5	57.8	67.2	60.9	67.2	53.1	51.5	50.7
SONG-D	9.4	9.4	12.5	15.6	20.3	12.5	10.9	9.3
NetDAP-P	7.8	7.9	7.8	12.5	18.7	11.2	9.4	9.4
NetDAP-N	6.3	7.8	7.8	9.4	11.1	7.8	6.4	6.2
NetDAP	6.3	6.4	7.9	7.8	9.4	7.8	6.3	6.2

S5	P1	P2	P3	P4	P5	P6	P7	P8
PAIG-DAP	219	266	359	219	344	235	218	187
NAM-APPRO	76.6	82.8	87.5	79.7	90.7	73.5	71.9	68.8
SONG-D	11.4	11.2	12.6	15.6	25.1	15.7	15.6	12.5
NetDAP-P	8.1	9.4	12.5	14.1	21.8	14.1	12.5	10.9
NetDAP-N	7.9	9.3	9.3	11.9	15.5	10.9	7.8	7.8
NetDAP	7.8	6.3	7.8	7.9	12.5	7.8	6.2	6.2

S6	P1	P2	P3	P4	P5	P6	P7	P8
PAIG-DAP	256	344	438	266	453	297	282	234
NAM-APPRO	79.7	86.4	92.2	78.1	87.5	79.7	75.8	71.9
SONG-D	11.1	11.3	14.1	15.6	21.9	17.2	15.7	14.1
NetDAP-P	9.4	9.4	12.5	14.1	21.8	15.7	15.6	12.5
NetDAP-N	7.8	9.4	10.9	10.9	11.4	9.4	7.8	7.8
NetDAP	7.8	7.8	7.9	9.4	9.3	9.4	7.6	7.5

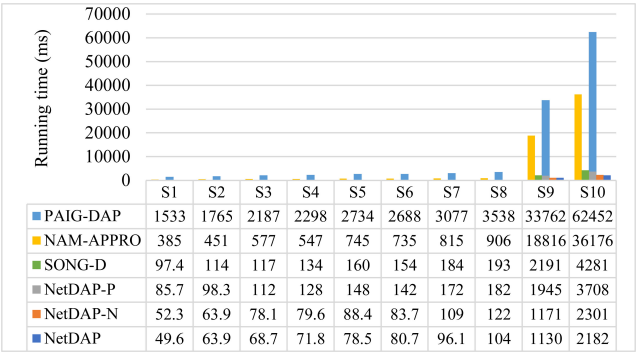
S7	P1	P2	P3	P4	P5	P6	P7	P8
PAIG-DAP	296	375	484	281	500	328	297	250
NAM-APPRO	90.6	98.4	103	89.1	112	86.2	82.8	81.2
SONG-D	12.5	14.1	17.2	20.3	29.7	18.8	17.2	15.6
NetDAP-P	9.4	10.5	12.5	18.7	26.6	15.7	14.1	12.5
NetDAP-N	9.3	9.5	11.1	14.1	18.6	13.4	11.5	10.5
NetDAP	9.2	9.4	10.9	12.5	15.6	12.5	10.9	9.4

S8	P1	P2	P3	P4	P5	P6	P7	P8
PAIG-DAP	343	406	546	328	547	360	328	297
NAM-APPRO	100	98.5	120	96.9	114	100	96.9	93.7
SONG-D	11.2	18.7	20.3	26.5	32.8	25.4	23.4	21.9
NetDAP-P	11.1	10.9	17.2	17.2	23.4	21.9	20.3	17.2
NetDAP-N	9.4	10.8	12.5	15.6	17.5	16.2	13.4	12.9
NetDAP	7.8	9.3	10.9	10.1	15.6	14.1	11.2	12.5

S9	P1	P2	P3	P4	P5	P6	P7	P8
PAIG-DAP	3344	4121	5379	3973	6385	3981	3801	3260
NAM-APPRO	886	917	1081	998	1083	968	936	821
SONG-D	222	328	339	351	402	293	265	234
NetDAP-P	145	139	196	267	465	237	265	193
NetDAP-N	120	120	145	150	185	203	150	143
NetDAP	118	129	136	151	178	160	156	147

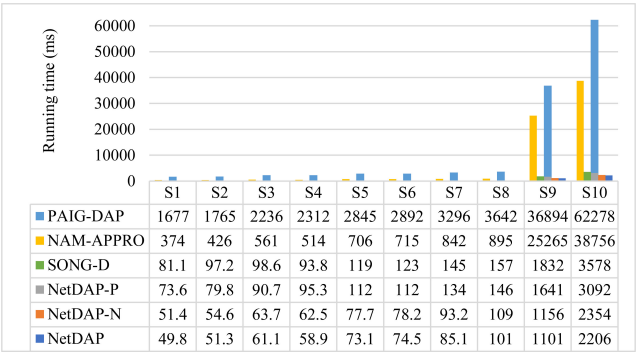
S10	P1	P2	P3	P4	P5	P6	P7	P8
PAIG-DAP	6230	8526	11130	7396	12117	7856	7224	6708
NAM-APPRO	1743	1879	2097	2016	2318	1898	1803	1635
SONG-D	527	480	879	635	843	653	615	570
NetDAP-P	387	387	568	603	821	632	547	474
NetDAP-N	334	362	525	564	801	532	479	404
NetDAP	297	309	446	412	456	375	365	312

Fig. 7: Comparison of the running time on $S1 - S8$ with $(\delta = 1, \gamma = 2)$



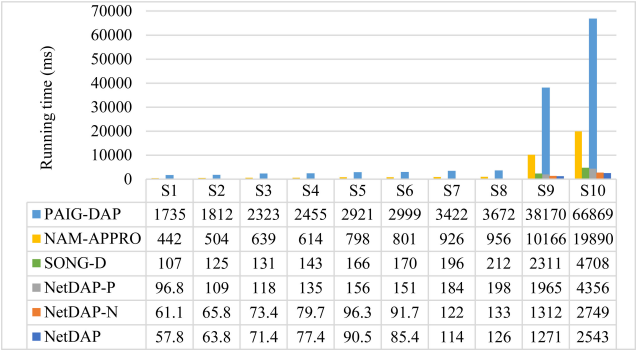
	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10
PAIG-DAP	1533	1765	2187	2298	2734	2688	3077	3538	33762	62452
NAM-APPRO	385	451	577	547	745	735	815	906	18816	36176
SONG-D	97.4	114	117	134	160	154	184	193	2191	4281
NetDAP-P	85.7	98.3	112	128	148	142	172	182	1945	3708
NetDAP-N	52.3	63.9	78.1	79.6	88.4	83.7	109	122	1171	2301
NetDAP	49.6	63.9	68.7	71.8	78.5	80.7	96.1	104	1130	2182

Fig. 8: Comparison of the total running time with $(\delta = 2, \gamma = 2)$



	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10
PAIG-DAP	1677	1765	2236	2312	2845	2892	3296	3642	36894	62278
NAM-APPRO	374	426	561	514	706	715	842	895	25265	38756
SONG-D	81.1	97.2	98.6	93.8	119	123	145	157	1832	3578
NetDAP-P	73.6	79.8	90.7	95.3	112	112	134	146	1641	3092
NetDAP-N	51.4	54.6	63.7	62.5	77.7	78.2	93.2	109	1156	2354
NetDAP	49.8	51.3	61.1	58.9	73.1	74.5	85.1	101	1101	2206

Fig. 9: Comparison of the total running time with $(\delta = 1, \gamma = 3)$

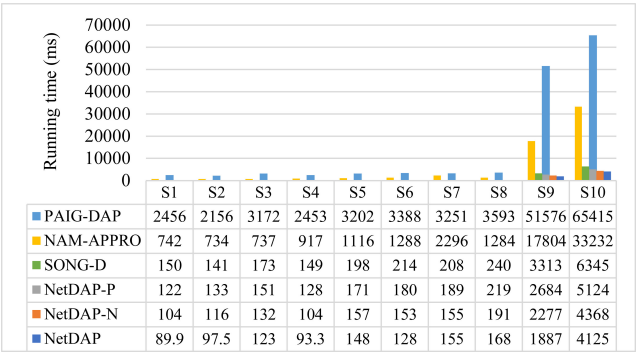


	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10
PAIG-DAP	1735	1812	2323	2455	2921	2999	3422	3672	38170	66869
NAM-APPRO	442	504	639	614	798	801	926	956	10166	19890
SONG-D	107	125	131	143	166	170	196	212	2311	4708
NetDAP-P	96.8	109	118	135	156	151	184	198	1965	4356
NetDAP-N	61.1	65.8	73.4	79.7	96.3	91.7	122	133	1312	2749
NetDAP	57.8	63.8	71.4	77.4	90.5	85.4	114	126	1271	2543

Fig. 10: Comparison of the total running time with $(\delta = 2, \gamma = 3)$

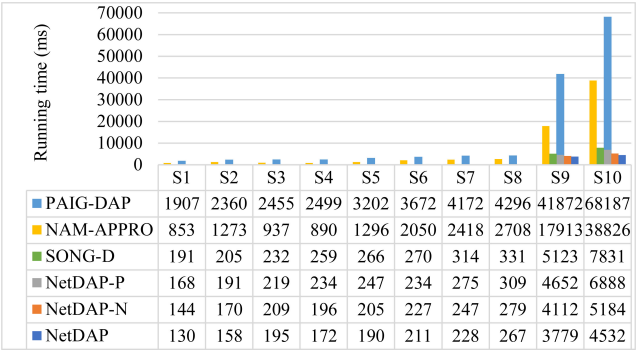
Table 5: The total number of occurrences with different parameters

Parameters	$S1$	$S2$	$S3$	$S4$	$S5$	$S6$	$S7$	$S8$	$S9$	$S10$
$(\delta = 2, \gamma = 2)$	1257	2457	1801	471	2144	2484	2552	4078	50336	68477
$(\delta = 1, \gamma = 3)$	2901	5795	3426	1143	4973	4818	5922	8850	98954	153101
$(\delta = 2, \gamma = 3)$	4157	7800	6051	1708	7525	8119	9192	13114	157233	237690
$(\delta = 1, \gamma = 4)$	5317	9493	5371	2120	9143	7716	11401	14085	159629	283079
$(\delta = 2, \gamma = 4)$	10504	18453	15414	4507	19921	21110	24075	30746	369875	606205



	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10
PAIG-DAP	2456	2156	3172	2453	3202	3388	3251	3593	51576	65415
NAM-APPRO	742	734	737	917	1116	1288	2296	1284	17804	33232
SONG-D	150	141	173	149	198	214	208	240	3313	6345
NetDAP-P	122	133	151	128	171	180	189	219	2684	5124
NetDAP-N	104	116	132	104	157	153	155	191	2277	4368
NetDAP	89.9	97.5	123	93.3	148	128	155	168	1887	4125

Fig. 11: Comparison of the total running time with $(\delta = 1, \gamma = 4)$



	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10
PAIG-DAP	1907	2360	2455	2499	3202	3672	4172	4296	41872	68187
NAM-APPRO	853	1273	937	890	1296	2050	2418	2708	17913	38826
SONG-D	191	205	232	259	266	270	314	331	5123	7831
NetDAP-P	168	191	219	234	247	234	275	309	4652	6888
NetDAP-N	144	170	209	196	205	227	247	279	4112	5184
NetDAP	130	158	195	172	190	211	228	267	3779	4532

Fig. 12: Comparison of the total running time with $(\delta = 2, \gamma = 4)$

From Figs. 7-12, we see that NetDAP outperforms all competitive algorithms. Details are as follows.

– NetDAP outperforms NetDAP-N and NetDAP-P. For example, in Fig. 12, the running time of NetDAP, NetDAP-N, and NetDAP-P for all patterns in sequence $S9$ are 3779 ms, 4112 ms, and 4652 ms, respectively. Thus, NetDAP is about 1.1 times faster than NetDAP-N and 1.2 times faster than NetDAP-P. All other experiments show the similar results. The main reasons are shown

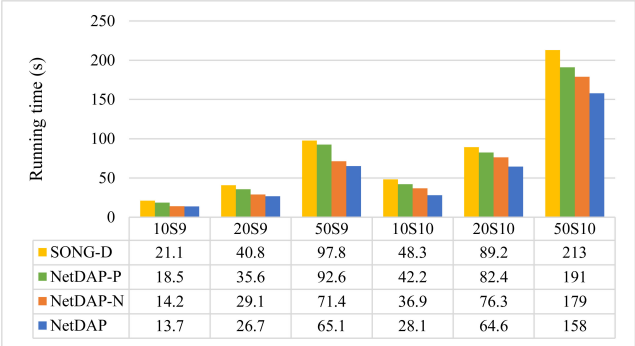
as follows. NetDAP is faster than both NetDAP-N and NetDAP-P, since NetDAP employs two pruning strategies, while NetDAP-N and NetDAP-P have only one. All these experiments verify that both node and parent-child relationship pruning strategies are effective. Meanwhile, node pruning strategy is more effective than parent-child relationship pruning strategy. Thus, NetDAP-N is faster than NetDAP-P. Furtherly, NetDAP is faster than NetDAP-N and NetDAP-P.

- NetDAP outperforms SONG-D. For example, in Fig. 11, the running time of NetDAP and SONG-D for all patterns in sequence $S2$ are 97.5ms and 141ms, respectively. Thus, NetDAP is about 1.4 times faster than SONG-D. The reason is that NetDAP employs two effective pruning strategies, while SONG-D has no pruning strategy. Therefore, NetDAP is faster than SONG-D. Meanwhile, the essential principles of NetDAP and SONG-D are similar, where SONG-D creates a single-root Nettree, while NetDAP creates a single-leaf Nettree. Thus, both NetDAP and SONG-D are faster than NAM-APPRO and PAIG-DAP, since the two algorithms avoid a lot of redundant computation.
- NetDAP outperforms NAM-APPRO and PAIG-DAP. For example, in Fig. 10, the running time of NetDAP, NAM-APPRO, and PAIG-DAP for all patterns in sequence $S3$ are 71.4 ms, 639 ms, and 2323 ms, respectively. The reason is shown as follows. NetDAP, NAM-APPRO, and PAIG-DAP employ single-leaf Nettree, Nettree, and multi-dimensional array data structure, respectively. PAIG-DAP employs multi-dimensional array to deal the problem which is a sparse array. Thus, PAIG-DAP is less efficient than NetDAP and NAM-APPRO. NAM-APPRO has to tackle with the length constraints and approximate constraints at the same time, while NetDAP only tackles with the approximate constraints. Thus, NAM-APPRO is more complex than NetDAP. Hence, NetDAP is faster than NAM-APPRO.

In summary, NetDAP has better performance than all competitive algorithms.

5.4 Running time performance in large data set

To further evaluate the running time performance in large data set, in this subsection, we construct larger data sets which are 10 times, 20 times, and 50 times length of S9 and S10, and named as 10S9, 20S9, 50S9, 10S10, 20S10, and 50S10, respectively. The length of 50 times of S10 reaches about 10Mb since the length of S10 is 204,482 byte. SONG-D, NetDAP-N, and NetDAP-P are selected as competitive algorithms. In the experiments, parameters ($\delta = 1, \gamma = 2$) are selected. The comparison results are shown in Fig. 13.



	10S9	20S9	50S9	10S10	20S10	50S10
SONG-D	21.1	40.8	97.8	48.3	89.2	213
NetDAP-P	18.5	35.6	92.6	42.2	82.4	191
NetDAP-N	14.2	29.1	71.4	36.9	76.3	179
NetDAP	13.7	26.7	65.1	28.1	64.6	158

Fig. 13: Comparison of the total running time with $(\delta = 1, \gamma = 2)$

From Fig. 13, we know that NetDAP is about 1.3 times faster than SONG-D. For example, the running time of NetDAP and SONG-D in 50S10 is 203 s and 158 s, respectively. These experimental results are consistent with the short sequence results. It is worth noting that SONG-D is an improved algorithm of SONG [20] which was proposed for solving approximate PM problem with Hamming distance. To handle with (δ, γ) distance, SONG-D is proposed based on SONG. SONG-D has no special designed pruning strategy since it is not a specially designed algorithm. However, NetDAP is a specially designed algorithm, which equips with node pruning and parent-child relationship pruning strategies that can avoid creating the invalid nodes and invalid parents, thus improving the running speed. Therefore, NetDAP runs about 1.3 times faster than SONG-D both in short sequences and in large data sets. Meanwhile, NetDAP-N and NetDAP-P run slower than NetDAP and faster than SONG-D since these two algorithms employ only one pruning strategy. These experimental results validate the effectiveness of the two pruning strategies.

5.5 Performance evaluations with respect to different parameters

In this subsection, we will evaluate the performance with respect to different parameters.

- **Approximate constraint.** From Tables 4 and 5, we can know when δ increases, the number of occurrences increases less. When parameters are $(\delta = 1, \gamma = 2)$, there are 3612 occurrences in $S8$, while when parameters are $(\delta = 2, \gamma = 2)$, there are 4078 occurrences. Hence, the growth rate $(N(S, P, \delta_1, \gamma)/N(S, P, \delta_2, \gamma))$ is about $4078/3612 = 1.12$. However, when parameters are $(\delta = 1, \gamma = 2)$, there are 1055 occurrences in $S1$, while when parameters are $(\delta = 1, \gamma = 3)$, there are 2901 occurrences. Hence, the growth rate is $2901/1055 = 2.74$. Similar phenomena can be found in other instances. Therefore, threshold γ affects the result more than threshold δ.
- **Sequence length.** The growth rate of running time is almost the same as that of sequence length. From Table 2, we know that the length of $S1$ to $S8$ increases gradually and the length of $S8$ is about 2 times that of $S1$. To further clarify the impact of large dataset on running time, the length of $S9$ and $S10$ are about 27 times and 51 times that of $S1$, respectively. From Fig. 8, NetDAP takes

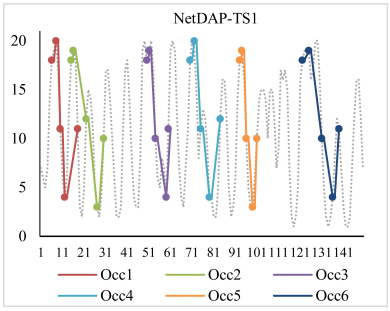
49.6ms in $S1$, 104ms in S8, 1130ms in S9, and 2182ms in $S10$ with parameters $(\delta = 2, \gamma = 2)$. Thus, the running time on $S8$, $S9$, and $S10$ are about 2 times, 23 times, and 44 times that of $S1$, respectively. Similar phenomena can be found in other experiments. Hence, these experimental results validate that the running time of NetDAP is linear correlated with sequence length. This is consistent with the time complexity of NetDAP. Meanwhile, it is easy to notice that NetDAP is remarkable faster than other competitive algorithms especially in large datasets.

- **Pattern length.** From Table 3, the length of $P4$ is shorter than that of $P5$. From Fig. 7, we find that the running time of $P4$ is shorter than that of $P5$ in all sequences, since the length of $P4$ is shorter than that of $P5$. For example, NetDAP takes 6.2ms on instance $P4-S1$, while 10.1ms on instance $P4-S8$ with parameters (δ=1, γ=2). Therefore, the running time is positively correlated to the pattern length.
- **Gap constraint.** From Table 3, the maximal gap of $P6$ to $P8$ increases gradually. From Fig. 7 we know that NetDAP runs the fastest in $P8$, since the maximal gap of $P8$ is the smallest. For example, NetDAP takes 4.5ms on instance $P8-S1$, while 7.8ms on instance $P6-S1$ with parameters $(\delta = 1, \gamma = 2)$. Hence, the running time is positively correlated to the maximal gap, i.e. the larger the gap, the longer the running time, and vice versa.
- δ**.** Figs. 9 and 10 have the same parameter δ while different parameter γ. We notice that with the increasing δ, the running time also increases. For example, NetDAP takes 49.8 ms and 57.8 ms in S1 with δ= 1 and δ=2, respectively. This phenomenon can also be found in Figs. 11 and 12. The reason lies that if parameter γ is the same, the greater δ is, the more the nodes will be created. Therefore, the running time increases with the increasing δ.
- γ**.** From Figs. 8, 10, and 12, we know that with the increasing γ, the running time also increases. For example, when γ= 2, γ=3, and γ=4, NetDAP takes 104 ms, 126 ms, and 267 ms, respectively, in $S8$. The reason lies that with the increase of γ, the time complexity of $NR(n_j^i, d)$ increases. Hence, the running time increases. Thus, the running time is positively correlated to γ.

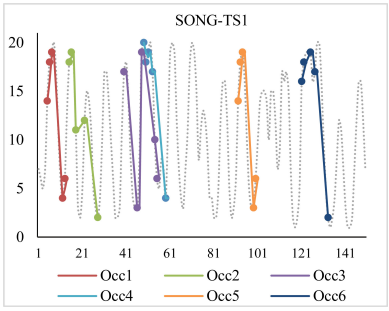
In all, we know that the running time of NetDAP is positively correlated to m, n, g, and γ, which is consistent with the theoretical analysis that the time complexity of NetDAP is $O(n * m^2 * g^2 * \gamma)$. More importantly, all competitive algorithms show similar phenomena with NetDAP in the above parameters.

5.6 Approximate matching performance

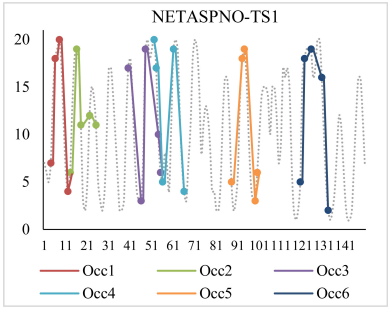
In this subsection, we select four time series to evaluate the approximate matching performance with (δ, γ) distance and Hamming distance. Parameters $(\delta = 1, \gamma = 2)$ mean that the approximate matching allows up to two different characters. Thus, the Hamming distance $d = 2$. Similarly, parameters $(\delta = 2, \gamma = 3)$ mean that the approximate matching allows up to three different characters, i.e. $d = 3$. Four group experiments are $P9$ in TS1 with $(\delta = 1, \gamma = 2)$ and $d = 2$, $P10$ in TS2 with $(\delta = 2, \gamma = 3)$ and $d = 3$, $P9$ in TS3 with $(\delta = 1, \gamma = 2)$ and $d = 2$, and $P10$ in TS4 with $(\delta = 2, \gamma = 3)$ and $d = 3$. SONG [20] and NETASPNO [35] are selected. The results are shown in Figs. 14-17.



(a) NetDAP with ($\delta = 1, \gamma = 2$)



(b) SONG with$d$=2

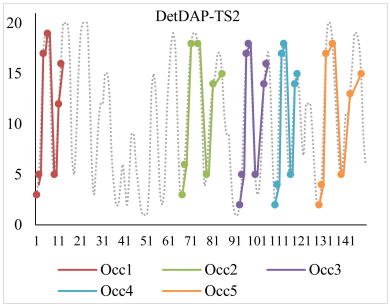


(c) NETASPNO with d=2

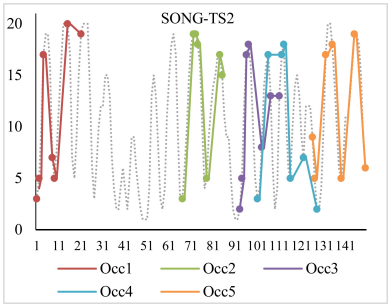
Fig. 14: Comparison of matching results of $P9$ in TS1

From Figs. 14-17, we can say that NetDAP has better approximate matching performance than SONG and NETASPNO. For example, Occ2 in Fig. 14 (a) is subsequence "RSLCJ". From Table 3, we know that $P9$=R[0,5]S[0,5]K[0,5]C[0,5]K. Without considering the gaps, $P9$ = RSKCK. Subsequence "RSLCJ" is an approximate occurrence of pattern $P9$ with ($\delta = 1, \gamma = 2$), since its δ and γ distances are 1 and 2, respectively, i.e. $|R - R| = 0, |S - S| = 0, |L - K| = 1, |C - C| = 0$, and $|J - K| = 1$ and $0 + 0 + 1 + 0 + 1 = 2$. Thus, from Fig. 14 (a), we know that the trend of Occ2 is similar with that of pattern $P9$. However, Occ2 in Fig. 14 (b) and (c) are subsequence "RSKLB" and "FSKLK", respectively. The Hamming distance between "RSKLB" and $P9$ is 2, since there are two characters in "RSKLB" different from "RSKCK". Similarly, The Hamming distance between "FSKLK" and $P9$ is also 2. Thus, the trends of Occ2 of SONG in Fig. 14 (b) and NETASPNO in Fig. 14 (c) deviate significantly from that of pattern $P9$. Similarly, similar phenomena can be found in Figs. 15-17.

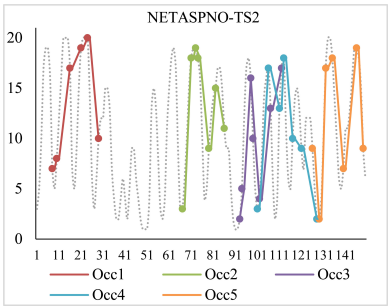
These experiments demonstrate that the approximate PM with (δ, γ) distance outperforms the approximate PM with Hamming distance.



(a) NetDAP with $(\delta = 2, \gamma = 3)$

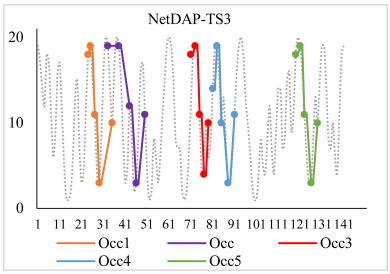


(b) SONG with d=3

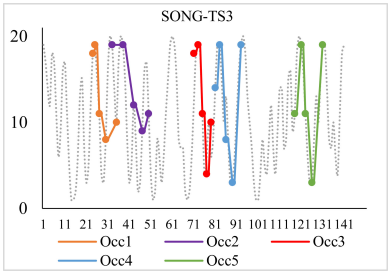


(c) NETASPNO with d=3

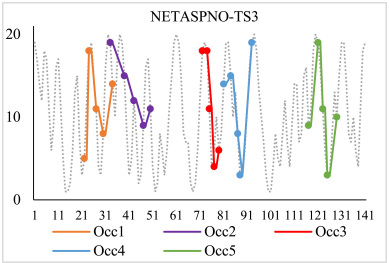
Fig. 15: Comparison of matching results of $P10$ in TS2



(a) NetDAP with $(\delta = 1, \gamma = 2)$

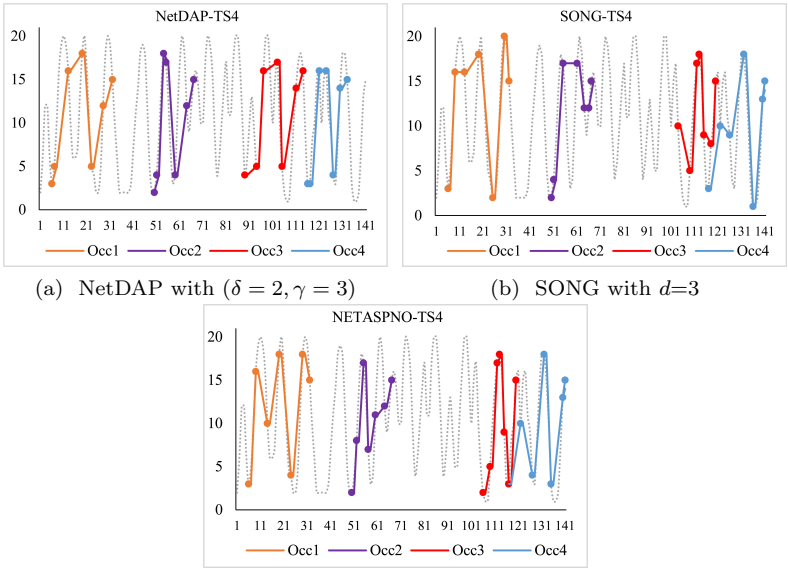


(b) SONG with d=2



(c) NETASPNO with d=2

Fig. 16: Comparison of matching results of $P9$ in TS3



(a) NetDAP with ($\delta = 2, \gamma = 3$) (b) SONG with d=3

(c) NETASPNO with d=3

Fig. 17: Comparison of matching results of $P10$ in TS4

6 Conclusion

Approximate PM with gap constraints mainly adopts the Hamming distance to measure the approximation degree which only reflects the number of different characters between two sequences, but ignores the distance between different characters. To solve this problem, in this paper, we address (δ, γ) approximate PM with length constraints, where the maximal distance between two characters is not greater than the local threshold δ, and the sum of all δ distances is also not greater than the global threshold γ. Therefore, our method has better performance than the traditional approximate PM with Hamming distance, since our method can avoid matching two characters with significant differences. To tackle (δ, γ) approximate PM with length constraints, we propose an effective online algorithm, named NetDAP, which adopts a special designed data structure, named the approximate single-leaf Nettree with (δ, γ) distance. NetDAP employs two effective pruning strategies which can prune useless nodes and the parent-child relationships. The space and time complexities of NetDAP are $O(\gamma * g * m^2)$ and $O(\gamma * g^2 * m^2 * n)$, respectively, where γ, g, m, and n are the similarity threshold, the maximal pattern gap, the pattern length and the sequence length, respectively. Extensive experimental results on real protein data and time series verify the correctness and efficiency of NetDAP.

Although the NetDAP algorithm proposed in this paper has better performance, there are still some shortcomings. (δ, γ) distance only considers the distance between different characters, which ignores the trend of pattern and loses valuable information. Therefore, a suitable approximate constraint distance is worth further study. Furthermore, as mentioned in Relate Work section, there are three types

of conditions: no-condition, nonoverlapping condition, and one-off condition. This paper focuses on no-condition which is the key issue of the nonoverlapping and one-off conditions. Our future work will focus on the nonoverlapping and one-off (δ, γ)-approximate pattern matching. We will also mine frequent patterns on time series employing (δ, γ) approximate PM. In this paper, we propose the NetDAP algorithm which employs node and parent-child relationship pruning strategies. In the future, we will explorer more applications to report on the benefits of NetDAP.

Acknowledgement

This work was partly supported by National Natural Science Foundation of China (61976240, 61571180, 917446209), National Key Research and Development Program of China(2016YFB1000901), and Graduate Student Innovation Program of Hebei Province (CXZZSS2019035).

References

1. Fernau H, Manea F, Merca R, Schmid M L (2020) Pattern matching with variables: Efficient algorithms and complexity results. ACM Transactions on Computation Theory (TOCT) 12(1): 1-37
2. Sotoodeh M, Tajeripour F, Teimori S, Jorgensen K (2018) A music symbols recognition method using pattern matching along with integrated projection and morphological operation techniques. Multimedia Tools and Applications 77(13): 16833-16866.
3. Navarro G (2014) Spaces, trees, and colors: The algorithmic landscape of document retrieval on sequences. ACM Computing Surveys (CSUR) 46(4): 52
4. Chen X, Rao Y, Xie H, Wang FL, Zhao Y, Yin J (2019) Sentiment classification using negative and intensive sentiment supplement information, Data Science and Engineering 4 (2): 109-118
5. Hu H, Zheng K, Wang X, Zhou A (2014) GFilter: A general gram filter for string similarity search. IEEE Transactions on Knowledge and Data Engineering 27(4): 1005-1018
6. Aldwairi M, Hamzah A Y, Jarrah M (2019) MultiPLZW: A novel multiple pattern matching search in LZW-compressed data. Computer Communications 145: 126-136
7. Choi B, Chae J, Jamshed M, Park K, Han D (2016) DFC: Accelerating string pattern matching for network applications. USENIX Symposium on Networked Systems Design and Implementation 2016: 551-565
8. Jiang H, Chen X, He T, Chen Z, Li X (2018) Fuzzy clustering of crowdsourced test reports for apps, ACM Transactions on Internet Technology (TOIT) 18(2): 1-28
9. Le H, Prasanna V K (2012) A memory-efficient and modular approach for large-scale string pattern matching. IEEE Transactions on Computers 62(5): 844-857
10. Ghosh S, Li J, Cao L, Ramamohanarao K (2017) Septic shock prediction for ICU patients via coupled HMM walking on sequential contrast patterns. Journal of Biomedical Informatics 66: 19-31

11. Wu X, Zhu X, Wu G Q, Ding W (2014) Data mining with big data. IEEE Transactions on Knowledge and Data Engineering 26(1): 97107
12. Song W, Liu Y, Li J (2014) Mining high utility itemsets by dynamically pruning the tree structure. Applied Intelligence 40 (1):2943
13. Wu M, Wu X (2019) On big wisdom. Knowledge and Information Systems 58(1): 1-8
14. Bille P, Fischer J, Grtz I L, Kopelowitz T, Sach B, Vildhj H W (2016) Sparse text indexing in small space. ACM Transactions on Algorithms (TALG) 12(3): 39
15. Gan W, Lin J C W, Fournier-Viger P, Chao H C, S.Yu P (2019) HUOPM: High-utility occupancy pattern mining. IEEE Transactions on Cybernetics DOI: 10.1109/TCYB.2019.2896267
16. Dong X, Qiu P, Lu J, Cao L (2019) Mining top-k useful negative sequential patterns via learning. IEEE Transactions on Neural Networks and Learning Systems 30(9): 2764-2778
17. Belhadi A, Djenouri Y, Lin JC, Cano A (2020) A general-purpose distributed pattern mining system. Applied Intelligence. DOI: 10.1007/s10489-020-01664-w
18. Bai L, Li Y, Liu J (2017) FSPTwigFast: Holistic twig query on fuzzy spatiotemporal XML data. Applied Intelligence 47(4): 1224-1239
19. Bouakkaz M, Ouinten Y, Loudcher S, Fournier-Viger P (2018) Efficiently mining frequent itemsets applied for textual aggregation. Applied Intelligence 48(4): 1013-1019
20. Wu Y, Tang Z, Jiang H, Wu X (2016) Approximate pattern matching with gap constraints. Journal of Information Science 42(5):639-658
21. Nip K, Wang Z, Xing W (2016) A study on several combination problems of classic shop scheduling and shortest path. Theoretical Computer Science 654: 175-187
22. Drory Retwitzer M, Polishchuk M, Churkin E, Kifer L, Yakhini Z, Barash D (2015) RNAPattMatch: A web server for RNA sequence/structure motif detection based on pattern matching with flexible gaps. Nucleic Acids Research 43(W1): W507-W512
23. Tan C D, Min F, Wang M, Zhang H R, Zhang Z H (2016) Discovering patterns with weak-wildcard gaps. IEEE Access 4: 4922-4932
24. Yen S J, Lee Y S (2013) Mining non-redundant time-gap sequential patterns. Applied Intelligence 39(4): 727-738
25. Li C, Yang Q, Wang J, Li M (2012) Efficient mining of gap-constrained subsequences and its various applications. ACM Transactions on Knowledge Discovery from Data (TKDD) 6(1): 2
26. Wu Y, Fu S, Jiang H, Wu X (2015) Strict approximate pattern matching with general gaps. Applied Intelligence 42(3): 566-580
27. Yang H, Duan L, Hu B, Deng S, Wang W, Qin P (2015) Mining top-k distinguishing sequential patterns with gap constraint. Journal of Software 26(11): 2994-3009
28. Wang H F, Duan L, Zuo J, Wang W, Li Z, Tang-Chin.J C (2016) Efficient mining of distinguishing sequential patterns without a predefined gap constraint. Chinese Journal of Computers 39(10): 1979-1991
29. Wu Y, Liu Y, Guo L, Wu X (2013) Subnettrees for strict pattern matching with general gaps and length constraints. Journal of Software 24(5): 915-932

30. Haapasalo T, Silvasti P, Sippu S, Soisalon-Soininen E (2011) Online dictionary matching with variable-length gaps. International Symposium on Experimental Algorithms. Springer, Berlin, Heidelberg 2011: 76-87
31. Shi Q, Shan J, Yan W, Wu Y, Wu X (2020) NetNPG: Nonoverlapping pattern matching with general gap constraints. Applied Intelligence DOI: 10.1007/s10489-019-01616-z
32. Sippu S, Soisalon-Soininen E (2013) Online matching of multiple regular patterns with gaps and character classes. International Conference on Language and Automata Theory and Applications. Springer, Berlin, Heidelberg 2013: 523-534
33. Wu Y, Shen C, Jiang H, Wu X (2017) Strict pattern matching under non-overlapping condition. Science China Information Sciences 60(1): 012101
34. Hu H, Wang H, Li J, Gao H (2016) An efficient pruning strategy for approximate string matching over suffix tree. Knowledge and Information Systems 49(1): 121-141
35. Wu Y, Li S, Liu J, Wu X (2018) NETASPNO: Approximate strict pattern matching under nonoverlapping condition. IEEE Access 6: 24350-24361
36. Arslan A N (2018) A fast algorithm for all-pairs Hamming distances. Information Processing Letters 139: 49-52
37. Bille P, Grtz I L, Vildhj H W, Wind DK (2012) String matching with variable length gaps. Theoretical Computer Science 443: 25-34.
38. Wu Y, Wang L, Ren J, Ding W, Wu X (2014) Mining sequential patterns with periodic wildcard gaps. Applied Intelligence 41(1):99-116
39. Wang X, Duan L, Dong G, Ye Z, Tang C (2014) Efficient mining of density-aware distinguishing sequential patterns with gap constraints. International Conference on Database Systems for Advanced Applications. Springer, Cham 372-387
40. Arslan A N, George B, Stor K (2015) New algorithms for pattern matching with wildcards and length constraints. Discrete Mathematics, Algorithms and Applications 7(3): 1550032
41. Liu N, Xie F, Wu X (2018) Multi-pattern matching with variable-length wildcards using suffix tree. Pattern Analysis and Applications 21(4): 1151-1165
42. Wu Y, Wang Y, Liu J, Yu M, Liu J, Li Y (2019) Mining distinguishing subsequence patterns with nonoverlapping condition, Cluster Computing 22 (3): 5905-5917
43. Liu H, Wang L, Liu Z, Zhao P, Wu X (2018) Efficient pattern matching with periodical wildcards in uncertain sequences. Intelligent Data Analysis 22(4): 829-842
44. Kim J, Eades P, Fleischer R, Hong S, Iliopoulos C S, Park K, Puglisi S J, Tokuyama T (2014) Order-preserving matching. Theoretical Computer Science 525: 68-79.
45. Crochemore M, Iliopoulos C S, Makris C, Rytter W, Tsakalidis A K, Tsichlas T (2002) Approximate string matching with gaps. Nordic Journal of Computing 9(1): 54-65
46. Navarro G, Raffinot M (2013) Fast and simple character classes and bounded gaps pattern matching, with applications to protein searching. Journal of Computational Biology 10(6): 903-23
47. Dong X, Gong Y, Cao L (2018) e-RNSP: An efficient method for mining repetition negative sequential patterns. IEEE Transactions on Cybernetics DOI: 10.1109/TCYB.2018.2869907

48. Wang R, Ji W, Liu M, Wang X, Weng J, Deng S, Gao S Y, Yuan C (2018) Review on mining data from multiple data sources. Pattern Recognition Letters 109: 120-128
49. Le T, Vo B, Fournier-Viger P, Lee M Y, Baik S W (2019) SPPC: A new tree structure for mining erasable patterns in data streams. Applied Intelligence 49(2): 478-495
50. Wu Y, Tong Y, Zhu X, Wu X (2018) NOSEP: Nonoverlapping sequence pattern mining with gap constraints. IEEE Transactions on Cybernetics 48(10):2809-2822
51. Min F, Zhang Z , Zhai W J, Shen R P (2020) Frequent pattern discovery with tri-partition alphabets. Information Sciences 507: 715-732.
52. Song W, Jiang B, Qiao Y (2018) Mining multi-relational high utility itemsets from star schemas, Intelligent Data Analysis 22 (1): 143-165
53. U Yun, H Nam, G Lee, E Yoon (2019) Efficient approach for incremental high utility pattern mining with indexed list structure. Future Generation Computer Systems 95, 221-239
54. Xie F, Wu X, Zhu X (2017) Efficient sequential pattern mining with wildcards for keyphrase extraction, Knowledge-Based Systems 115: 27–39
55. Guo D, Hu X, Xie F, Wu X (2013) Pattern matching with wildcards and gap-length constraints based on a centrality-degree graph. Applied Intelligence 39: 57-74
56. Wu Y, Zhu C, Li Y, Guo L, Wu X (2020) NetNCSP: Nonoverlapping closed sequential pattern mining. Knowledge-Based Systems
57. Fischer M J, Paterson M S (1974) String-matching and other products. Proceedings of the 7th SIAM ANS Complexity of Computation 1974:113-125
58. Manber U, BaezaYates R (1991) An algorithm for string matching with a sequence of don't cares. Information Processing Letters 37(3): 133-136
59. Min F, Wu X, Lu Z (2009) Pattern matching with independent wildcard gaps. Proceedings of the 8th International Conference on Pervasive Intelligence and Computing 2009: 194-199
60. Wu Y, Wu X, Min F, Li Y (2010) A Nettree for pattern matching with flexible wildcard constraints. 2010 IEEE International Conference on Information Reuse and Integration 2010: 109-114
61. Warmuth M K, David H (1984) On the complexity of iterated shuffle. Journal of Computer and System Sciences 28(3): 345-358
62. Guo D, Yuan E, Hu X (2016) Frequent pattern mining based on approximate edit distance matrix. IEEE First International Conference on Data Science in Cyberspace (DSC) 2016: 179-188
63. Min F, Wu Y, Wu X (2010) The Apriori property of sequence pattern mining with wildcard gaps. IEEE International Conference on Bioinformatics and Biomedicine Workshops 2010: 138-143
64. Lin J, Keogh E, Wei L, Lonardi S (2007) Experiencing SAX: A novel symbolic representation of time series. Data Mining and Knowledge Discovery 15(2):107-144