2010 22nd International Conference on Tools with Artificial Intelligence

A Nettree for Approximate Maximal Pattern Matching with Gaps and One-Off Constraint

Youxi Wu*†, Xindong Wu‡†, He Jiang§†, Fan Min¶
*School of Computer Science and Software, Hebei University of Technology, Tianjin 300130, China
†Department of Computer Science,University of Vermont, Burlington, VT 05405, USA
‡School of Computer Science and Information Engineering, Hefei University of Technology, Hefei 230009, China
§School of Software, Dalian University of Technology, Dalian 116621, China
¶School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 610054, China

***Abstract*—Recently, pattern matching with flexible gap constraints has attracted extensive attention especially in biological sequence analysis and mining patterns from sequences. An issue is to search Maximal Pattern Matching with Gaps and the One-Off Condition (MPMGOOC). Firstly, we introduce the concept of MPMGOOC. In order to solve the problem, we propose some special concepts of Nettree which is different from a tree in that a node may have more than one parent. Based on Nettree, an algorithm named Heuristic Search Occurrence (HSO) is proposed. The space and time complexities of the algorithm are $O(W*m*n)$ and $O(W*n*(n+m*m))$ respectively, where m, n, and W are the length of pattern P, sequence S and the maximal gap respectively. The comparison results show that HSO achieves better performance than a state-of-the-art algorithm in most cases of the real-world biological data testing.**

I. Introduction

Pattern matching with wildcards ("don't cares"), often marked as "*" (or "?", "ϕ", "$\sharp$") which can match any letter in a given alphabet [1], [2], [3], [4], [5], [6], [7], can be used in many fields such as biological sequence analysis[1], [2], text indexing[3] , time series data mining[4], stream data mining[5], [6], pattern mining [8], [9] and so on. There are two main types of research efforts for the problem of string matching with wildcards. The first is engaged to a fixed length of wildcards and the second focuses on the flexible gap constraints. Fischer and Paterson developed an algorithm for solving pattern matching in 1974 [7] in which the length of wildcards in a pattern was fixed. Recently, pattern matching with flexible gap constraints has attracted extensive attention [2], [8], [9], [10], [11]. He et al [2] aimed to solve the problem of frequent pattern mining without user specified gap constraints. Huang et al [8] considered the problem of mining a set of gap constrained sequential patterns across multiple sequences. Manber et al [9] studied an algorithm for string matching with only one flexible wildcard. Min et al [10] proposed an algorithm, PAIG-RST (reduced space and time), to compute the number of occurrences of pattern matching with multi-constraints. Chen et al [11] proposed an online algorithm, SAIL, to solve Maximal Pattern Matching with Gap and the One-Off Constraints (MPMGOOC) online. The one-off condition means that each sequence position can be used at most once. However, SAIL is not a complete off-line algorithm [8].

In this paper, we propose an off-line heuristic algorithm based on a new nonlinear structure Nettree to solve the problem of MPMGOOC. A Nettree is an extension of a tree because it has all concepts of a tree such as the root, leaf, parent, child, path and so on. In order to solve the problem, some special concepts of Nettree are brought forward, such as the root path number, leaf path number, root-leaf path number, common descendant, the number of paths from current node to roots, the min root, the max root, and so on. Then an algorithm named Heuristic Search Occurrence (HSO) is proposed to solve the problem. The main idea of HSO is to find the Approximate Optimization Parent (AOP) of the current node at each step in the process of searching an Occurrence with the One-Off Condition (OOOC). The space and time complexities of HSO are $O(W*m*n)$ and $O(W*n*(n+m*m))$ respectively, where m, n, and W are the length of pattern P, sequence S, and the maximal gap respectively.

II. Problem Definition

Definition 1: Assume a **pattern** $P = p_0[min_0, max_0]p_1\cdots[min_{j-1}, max_{j-1}]p_j\cdots[min_{m-2}, max_{m-2}]p_{m-1}$, a **sequence** $S = s_0s_1\cdots s_i\cdots s_{n-1}$, and two integer values, $MinLen$ and $MaxLen$, where m is the length of pattern P, $p_j \neq \phi$ $(0 < j \leq m-1)$, n is the length of sequence S, and $s_i \neq \phi$ $(0 \leq i \leq n-1)$. ϕ is a **wildcard** which can match any letter in a given alphabet. min_{j-1} and max_{j-1} are given integer values indicating the minimal and maximal length of wildcards between two given letters p_{j-1} and p_j respectively, where $0 \leq min_j \leq max_j$. $MinLen$ and $MaxLen$ are called **global length constraints**, and min_j and max_j are called **local length constraints** [11].

Definition 2: An **occurrence** of pattern P in sequence S is a sequence of position indices $a_0, \cdots, a_j, \cdots, a_{m-1}$ subject to the following equation.

$$\text{Subject to} \quad \begin{array}{c} p_j = s_{a_j} \\ min_{j-1} \leq a_j - a_{j-1} - 1 \leq max_{j-1} \\ MinLen \leq a_{m-1} - a_0 + 1 \leq MaxLen, \\ a_{j-1} < a_j \end{array} \tag{1}$$

1082-3409/10 $26.00 © 2010 IEEE
DOI 10.1109/ICTAI.2010.81

where $0 \leq j \leq m-1$ and $0 \leq a_j \leq n-1$.

Definition 3: Let set $T(S,P)$ be the set of all occurrences of P in S. The length of $T(S,P)$ is denoted by $|T(S,P)|$.

Definition 4: Given two occurrences $B =< b_0, \cdots, b_j, \cdots, b_{m-1} >$ and $C =< c_0, \cdots, c_k, \cdots, c_{m-1} >$, occurrence C is **related with** occurrence B if there exists $c_k = b_j$, otherwise occurrence C is **unrelated with** occurrence B. The number of occurrences Related with Occurrence B is denoted by $RO(B)$. Both occurrences B and C **contain position** b_j, where $0 \leq k < m$ and $0 \leq j < m$. The number of occurrences Containing Position i ($0 \leq i \leq |S|-1$) of P in S is denoted by $CP(i)$.

Definition 5: Given a pattern P and a sequence S, a **Pattern Matching with Gaps and the One-Off Condition (PMGOOC)** is a subset $T'(S,P) \subseteq T(S,P)$ such that every two occurrences in $T'(S,P)$ are unrelated with each other. The **Maximal Pattern Matching with Gaps and the One-Off Condition (MPMGOOC)** problem refers to the problem of searching a MPMGOOC with the largest number of occurrences in $T(S,P)$.

Definition 6: Let $B =< b_0, \cdots, b_j, \cdots, b_{m-1} >$ be an occurrence of P in S. Sequence $S^* = s_0^* s_1^* \cdots s_k^* \cdots s_{n-1}^*$ under the one-off condition of occurrence B (denoted by $S-B$) can be calculated as follows.

s_k^* is X if $k = b_j$, otherwise s_k^* is s_k, where X is unmatched.

Definition 7: Given an occurrence $B =< b_0, \cdots, b_j, \cdots, b_{m-1} >$ and a set $D = \{d_0, \cdots, d_{l-1}\}$. Occurrence B is **related with set** D if there exists $b_j = d_r$, where $0 < l$ and $0 \leq r \leq l$. The number of occurrences Related with Set D is denoted by $RS(D)$.

Definition 8: Assuming $e \notin D_1$ and set $D_2 = D_1 \cup \{e\}$, the number of occurrences **Increasing** from e (denoted by $I(e, D_1)$) can be calculated as $I(e, D_1) = RS(D_2) - RS(D_1)$.

In order to understand the problem and definitions easily, some examples are given as follows.

Example 1: Assume P=a[0,1]b[0,1]c, S=aabbcc, $MinLen$ =3, and $MaxLen$=5.

We know that $T(S,P)$ is $\{< 0,2,4 >, < 1,2,4 >, < 1,3,4 >, < 1,3,5 >\}$ and $|T(S,P)|$ is 4. The solution of MPMGOOC is $\{< 0,2,4 >, < 1,3,5 >\}$.

Example 2: Let us consider the same problem in example 1. Given two occurrences B=$< 0,2,4 >$, C=$< 1,3,4 >$, set D=$\{2,3\}$ and $i = 2$.

Occurrence C is related with occurrence B since both occurrences B and C contain position 4. Occurrences B and C are related with set D since occurrences B and C contain positions 2 and 3 respectively, and 2 and 3 are two elements of set D. We achieve that $RO(B)$=3, $RO(C)$=4, $CP(i)$=2, and $RS(D)$=4. Similarly, Let b=3 and $D_1 = \{2,4\}$. We achieve that $RS(D_1)$=3 and $I(b, D_1) = RS(\{2,3,4\}) - RS(\{2,4\})$ =1. Sequence S^* under the one-off condition of occurrence B is aXbXcX. Occurrence B is an optimal occurrence since we can verify that $RO(B)$ gets the minimal value.

III. NETTREE

Definition 9: A **Nettree** is a directed acyclic graph (DAG) with two kinds of edge labels, "parent-child" and "child-parent", where each node has zero or more children nodes and zero or more parent nodes. Furthermore, the children of each node have a specific order. A Nettree has the following four properties.

(1). A Nettree is an extension of a tree because it has similar concepts of a tree, such as the root, leaf, level, parent, child, and so on.

(2). A Nettree may have more than one root.

(3). Some nodes except roots in a Nettree may have more than one parent.

(4). Some nodes may occur more than once.

Definition 10: If all nodes occur only once, we can use the node name to denote each node. Otherwise if node i occurs more than once in different levels in a Nettree, the j^{th} level node i is denoted by n_j^i.

Definition 11: Node b is an **ancestor** of node c if node b is in a path between a root and node c. We define that a node is an ancestor of itself. The set of all ancestors of node c is denoted by $A(c)$.

Definition 12: Given a node set $D = \{d_0, \cdots, d_{l-1}\}$. The **common descendant** of D is defined as $Branch(D) = A(d_0) \bigcap A(d_1) \bigcap \cdots \bigcap A(d_{l-1})$.

Definition 13: The number of paths from node n_j^i to roots is called the **Root Path Number (RPN)** and denoted by $N_r(n_j^i)$.

Property 1: The RPN of a j^{th} ($j > 1$) level node n_j^i ($N_r(n_j^i)$) is the sum of RPNs of its parents

$$N_r(n_j^i) = \sum_{l=1}^{k} N_r(n_{j-1}^{i_l}), \tag{2}$$

where i_l is the l^{th} parent of node n_j^i and k is the number of parents of node n_j^i.

Definition 14: The number of paths from node n_j^i to the m^{th} level leaves is called the **Leaf Path Number(LPN)** and denoted by $N_l(n_j^i)$, where m is the depth of the Nettree. We define as $N_l(n_m^i)$=1.

Property 2: The LPN of node n_j^i ($N_l(n_j^i)$, where $1 \leq j \leq m-1$ and m is the depth of Nettree) is the sum of LPN of its children

$$N_l(n_j^i) = \sum_{l=1}^{h} N_l(n_{j+1}^{i_l}), \tag{3}$$

where i_l is the l^{th} child of node n_j^i and h is the number of children of node n_j^i.

Definition 15: The number of paths containing node n_j^i from roots to the m^{th} level nodes is called the **Root-Leaf Path Number(RLPN)** and denoted by $N_p(n_j^i)$.

Property 3: The RLPN of node n_j^i ($1 \leq j \leq m$) is the multiplication of its RPN and its LPN.

Property 4: The number of occurrences containing position i ($CP(i)$) is the sum of RLPNs of nodes n_j^i in each level.

Definition 16: The number of paths containing node i in the common descendants of set C is denoted by $pb(i, C)$.

In order to consider the global constraints, we define the min root and the max root of each node.

Definition 17: Given a node b, its **min root** is the minimum root name which can be reached by node b. Similarly, the **max root** of node b is the maximal root name which can be reached by node b. We define that a root is both the min root and the max root of itself.

Property 5: The min root and the max root of a node are the min root and the max root of its parents respectively.

A Nettree is shown in Fig. 1. Some nodes occur more than once. Nodes n_1^1, n_1^2, and n_1^3 are three roots of the Nettree. Nodes n_2^2, n_3^5, and n_3^6 are three leaves. Node n_2^3 has two parents (nodes n_1^1 and n_1^2). In Fig. 1, each edge is labeled either "parent-child" or "child-parent". If we only consider each kind of edge label, there is no cycle in the directed graph. So it is a kind of DAG with edge label. Otherwise there will be a cycle $\{n_1^2, n_2^3, n_3^6, n_2^4, n_1^2\}$. Another characteristic of Nettree is that there may be more than 1 root path from a root to a node in a Nettree.

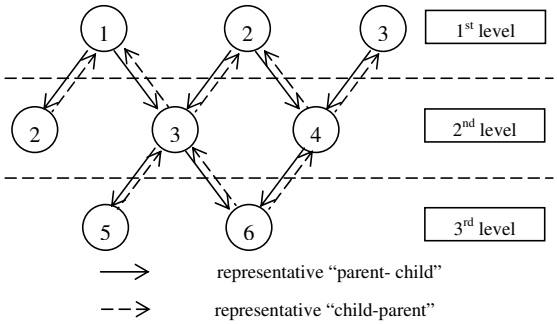

Fig. 1. A Nettree.

IV. Algorithm Design

A. Proposed algorithm

1) The main framework of HSO: The first step of HSO is the creation of a Nettree according to P and S, and the creation rules of Nettree are given as follows.

When a character s_i $(0 \leq i < n)$ arrives, we check whether s_i satisfies the following three rules or not. If yes, we create a node or an edge accordingly.

Rule 1. If $s_i = p_0$, node n_1^i will be created in the first level.

Rule 2. If $s_i = p_j$ and the distance between i and the j^{th} level node n_{j-1}^e satisfies the local constraints ($min_{j-1} \leq i-e-1 \leq max_{j-1}$) , node n_j^i will be created in the j^{th} level.

Rule 3. If the distance between nodes n_j^i and n_{j-1}^q satisfies the local constraints ($min_{j-1} \leq i-q-1 \leq max_{j-1}$), a parent-child relation between nodes n_j^i and n_{j-1}^q will be created.

The second step of HSO is to compute the min root and the max root for each node according to definition 17 in order to consider the global constraints. In order to find a solution, HSO iterates to call the algorithm of GSPO in order to find the AOP of the current node at each step in the process of searching an OOOC.

The framework of HSO

Input: $P=p_0[min_0,max_0]p_1 \cdots [min_{m-2},max_{m-2}]p_{m-1}$, $S = s_0 s_1 \cdots s_{L-1}$, *Minlen* and *Maxlen*
Output: Optimal occurrences (set *C*)

Algorithm:
1:create a Nettree(*NT*) according to *P* and *S*;
2:compute the min root and the max root for each node;
3: for *k*= the number of the m^{th} level leaves downto 1 step -1
4: *B*= GSPO (the k^{th} leaf);
5: $C=C \cup B$;
6: $S=S - B$;
7: compute RPN for each node according to new *S*;
8: next *k*
9:return *C*;

Algorithm GSPO

Input: Nettree *NT* and a leaf *f*
Output: An optimal occurrence (*B*)

Algorithm:
1: compute RPN for each node according to property 1;
2: compute LPN for each node according to property 2;
3: compute RLPN for each node according to property 3;
4: compute the containing position for each index *i* according to property 4;
5: *B*[*m*-1]= f;
6: for *j*=*m*-2 downto 0 step -1 do
7: compute *pb*(*x*,*B*) for each index *x* in branch with common descendants of *B*;
8: *r*=*B*[*j*+1].*number_of_parents* ;
9: *B*[*j*]= *B*[*j*+1].*parent*[*r*-1];
10: for *k*=*r*-2 downto 0 step -1 do
11: if *B*[*j*+1].*parent*[*k*] satisfied with global constrains then
12: if (*CP*(*B*[*j*])>*CP*(*B*[*j*+1].*parent*[*k*])) then *B*[*j*]= *B*[*j*+1].*parent*[*k*] ;
13: if (*CP*(*B*[*j*])=*CP*(*B*[*j*+1].*parent*[*k*])) and (*pb*(*B*[*j*+1].*parent*[*k*], *B*) >= *pb*(*B*[*j*], *B*) then *B*[*j*]= *B*[*j*+1].*parent*[*k*] ;
14: end if
15: end for
16: end for
17: return *B*

2) Algorithm GSPO: The main idea of GSPO is to find the AOP of the current node at each step in the process of searching an approximate optimization OOOC. In order to find occurrence B with the m^{th} level leaf n_m^l, GSPO computes RPN, LPN, and RLPN for each node at first. Then GSPO computes the containing position for each index according to RLPN. After that, GSPO iterates $m-1$ times to find the AOP of the current node. In order to find AOP n_j^x of node n_{j+1}^i, GSPO computes pb for each index in the common descendant at first, then GSPO uses values of CP and pb to find AOP n_j^x.

B. Complexity

The space complexity of HSO is $O(W * m * n)$ because the depth of the Nettree is m, each level has no more than n nodes, and each node has no more than W parents, where m, n, and W are the length of pattern P, sequence S, and the maximal gaps respectively.

The analysis of the time complexity of HSO is given as follows.

The time complexity of lines 1 and 2 of HSO is $O(W * m * n)$. We can know that the time complexity of GSPO is $O(W * m(n + m * m))$. So the time complexity of line 4 of HSO is $O(W * m(n + m * m))$. The time complexity of lines 5 and 6 is $O(m)$ because the length of the patterns is m. The time complexity of line 7 is $O(W * m * n)$. Therefore the time complexity of lines 3 through 8 of HSO is $O(W * m(n + m * m) * n/m) = O(W * n * (n + m * m)$ because the number of OOOCs is less than n/m. Hence the time complexity of HSO is $O(W * n * (n + m * m))$.

TABLE I
SEQUENCES OF REAL-WORLD BIOLOGICAL DATA

Number	Segment NO.	Locus	Its 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 II
PATTERNS

Number	Pattern	$MinLen$	$MaxLen$
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	11	41
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	24	57
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	21	101
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	27	73

V. EXPERIMENTS

In this section, real-world biological data are used to compare the ability of the algorithm of SAIL [11] with our algorithm. All experiments are conducted on a computer with Pentium (R) 4 CPU 3.40GHZ and 1.0 GB of RAM, Windows XP. The source codes of SAIL and HSO can be downloaded from [12].

The H1N1 (Swine Flu) 2009 virus was isolated during human swine flu outbreak of 2009. The biological sequences of the H1N1 2009 virus are downloaded from the National Center for Biotechnology Information website[13]. There were many candidates and segments and we selected all segments 1 to 8 of a newer result (A/Managua/1455.01/2009(H1N1)), which were published on March 26, 2010[14]. These segment sequences (see Table 1.) are used as testing sequences.

Min et al [10] gave some patterns in their research work. Since P5 of his work cannot be applied in DNA sequences, we select all other four patterns of his work (P1 to P4) as our test patterns. All testing patterns are shown in Table 2.

The time cost of all 32 instances of these 2 algorithms is less than 1 second. The results of SAIL and HSO are shown in Table 3. Comparison results show that HSO gets 27 results which are better than or equal to that of SAIL. So HSO gets better performance than SAIL in most cases of the real-world biological data testing.

TABLE III
BIOLOGICAL DATA TESTING RESULTS

Pattern	Algorithm	s1	s2	s3	s4	s5	s6	s7	s8
P1	SAIL	13	9	10	15	11	5	3	3
	HSO	13	9	10	15	11	5	3	3
P2	SAIL	66	69	59	54	42	39	31	27
	HSO	67	71	62	54	42	41	33	28
P3	SAIL	66	69	66	54	45	42	33	28
	HSO	64	70	68	52	43	43	33	26
P4	SAIL	49	50	49	40	32	31	24	20
	HSO	51	58	52	46	37	30	26	21

VI. CONCLUSIONS

The main contribution of this paper is to propose a new nonlinear structure, Nettree. A Nettree is different from a tree in that a node may have more than one parent. Since it is a new data structure, it has many special concepts and properties. In this paper, some special concepts and properties of the Nettree have been put forward such as RPN, LPN, RLPN, the min root, the max root, and so on. A heuristic algorithm HSO based on the Nettree has been proposed to solve the problem of MPMGOOC. HSO achieves better performance than SAIL in most cases of the real-world biological data testing. The time and space complexities of HSO are $O(W*n*(n+m*m))$ and $O(W*m*n)$ respectively, where m, n, and W are the length of pattern P, sequence S, and the maximal gap respectively. We believe that the Nettree will attract more theoretical studies.

ACKNOWLEDGEMENTS

This research is supported by the National Natural Foundation of China (NSFC) under grant No. 60828005.

REFERENCES

[1] Cole, J. R., Chai, B., Farris, R. J., Wang, Q., Kulam, S. A., McGarrell, D. M., Garrity, G. M., and Tiedje, J. M., The Ribosomal Database Project (RDP-II): sequences and tools for high-throughput rRNA analysis, Nucleic Acids Research, 2005, Vol. 33, pp. 294-296.

[2] He, Y., Wu, X., Zhu, X. and Arslan, AN, Mining Frequent Patterns with Wildcards from Biological Sequences, Proceedings of the 2007 IEEE International Conference on Information Reuse and Integration (IEEE IRI-07), 2007, pp. 329-334.

[3] Cole, R., Gottlieb, L.A., Lewenstein, M.: 2004. Dictionary matching and indexing with errors and don't cares. In: Proceedings of the 36th ACM Symposium on the Theory of Computing. ACM Press, New York, NY, USA, 91-100.

[4] Faloutsos, C., Ranganathan, M., and Manolopoulos, Y., Fast subsequence matching in time-series databases, Proceedings of ACM-SIGMOD International Conference on Management of Data (SIGMOD). 1994, Vol 23 , Issue 2, pp. 419 - 429.

[5] Fan, W., Wang, H., Yu, P. S., System and method for tree structure indexing that provides at least one constraint sequence to preserve query-equivalence between xml document structure match and subsequence match, United States Patent 7475070. 2009.

[6] Moon, Y. S., Efficient stream sequence matching algorithms for handheld devices on time-series stream data, Proceedings of the 24th IASTED International Conference on Databases and Applications. 2006, pp. 44-49.

[7] Fischer, M.J., Paterson, M.S.: 1974. String matching and other products. In: Karp RM(ed) Complexity of computation, Vol 7. Massachusetts Institute of Technology, Cambridge, MA, USA, 113-125.

[8] Huang, Y., Wu, X., Hu, X., Xie, F., Gao, J., Wu, G., Mining Frequent Patterns with Gaps and One-off Condition, Proceedings of the 12th IEEE International Conferences on Computational Science and Engineering. 2009, pp. 180-186.

[9] Manber U, Baeza-Yates R (1991) An algorithm for string matching with a sequence of don't cares. Inf. Proc. Lett. 37(3):133-136.

[10] Min, F., Wu, X., Lu, Z., Pattern matching with independent wildcard gaps. Proceedings of the 8th International Conference on Pervasive Intelligence and Computing. 2009, pp. 194-199.

[11] Chen, G., Wu, X., Zhu, X., Arslan, A. N., and He, Y.: 2006. Efficient string matching with wildcards and length constraints. Knowledge and Information Systems, vol. 10, no. 4, 399-419.

[12] http://www.cems.uvm.edu/ ∼youxiwu/

[13] http://www.ncbi.nlm.nih.gov/

[14] http://www.ncbi.nlm.nih.gov/genomes/FLU/SwineFlu.html