

Approximate Matching in Genomic Sequence Data



Xia Cao

Department of Computer Science
National University of Singapore

Introduction

- Approximate sequence matching is one of the classic problems in computer science.
- Many approaches have been developed for approximate sequence matching.
 - The fundamental one is the Smith-Waterman alignment algorithm ($O(mn)$).

Problems

- Similarity search in DNA sequence database
- DNA sequence approximate join
- Protein sequence subcellular localization prediction

DNA Sequence Similarity Search



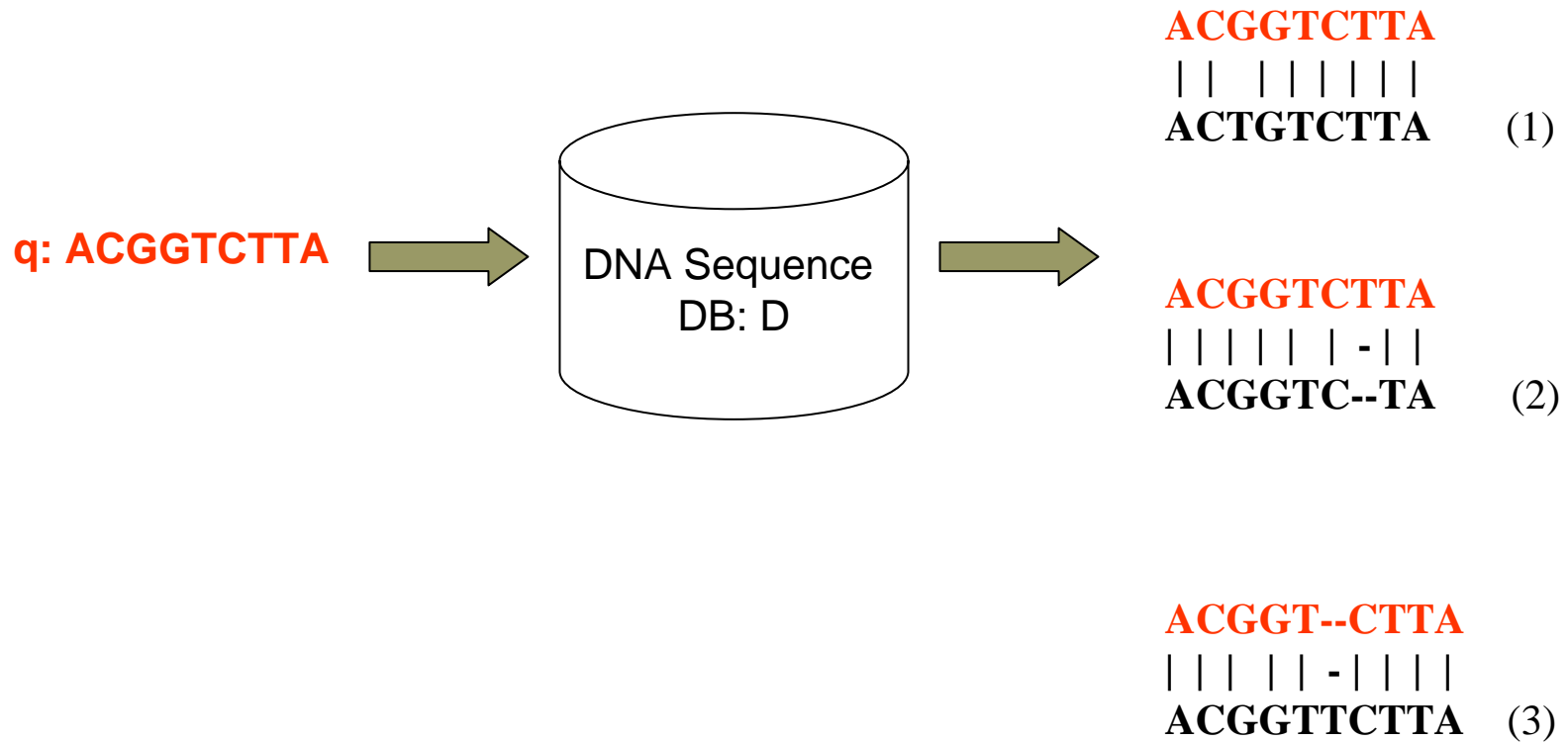
Problem I

Problem Definition

□ Problem:

- Given the length L and edit distance threshold θ , find all subsequences S in sequence database D which have length $|S| \geq L$ and $\text{edit}(S, Q') \leq \theta$ for subsequence Q' in query sequence Q .
 - Seed detection: Given the length ω and edit distance ε , find all the segments (seeds) s_i with length ω in D which meet $\text{edit}(s_i, q_j) \leq \varepsilon$ for the query segments q_j with length ω in Q .
 - Seed extension

Problem Definition (Cont.)



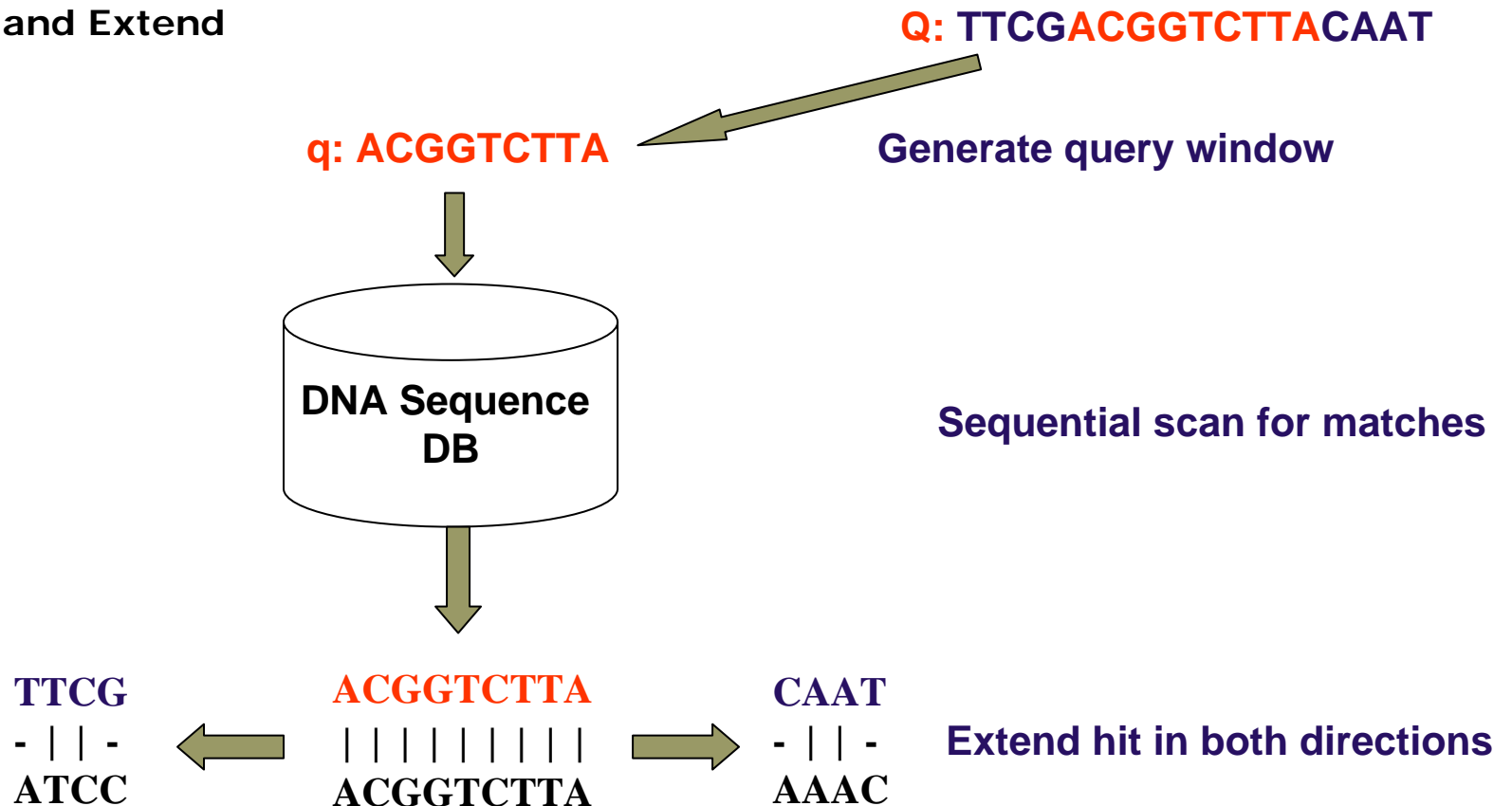
Related Work

- Scan based search
 - BLAST
 - FASTA
 - PatternHunter

- Index based search
 - Suffix tree or array: Quasar, Oasis
 - Inverted file index: CAFÉ
 - ed-tree
 - Multi-dimensional indexing approaches: wavelet-based method (FV), SST
 - Hash based approach: SSAHA

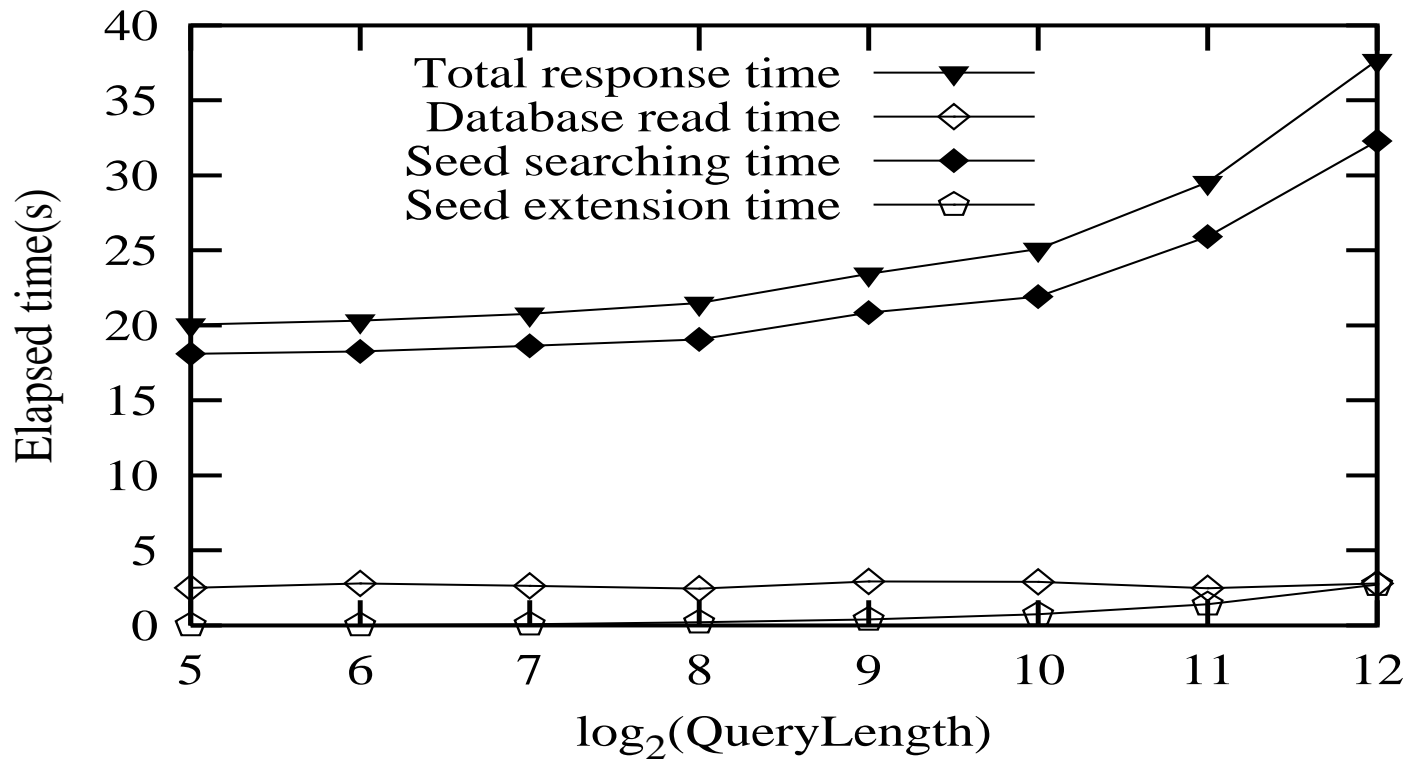
BLASTn

- BLASTn is the most popular tool for DNA sequence search
- Seed-based search tool
- Hit and Extend



BLASTn (Cont.)

Blast's Search in Genbank sequences (DNA)



Breakdown BLAST'S Search Time

Solution 1



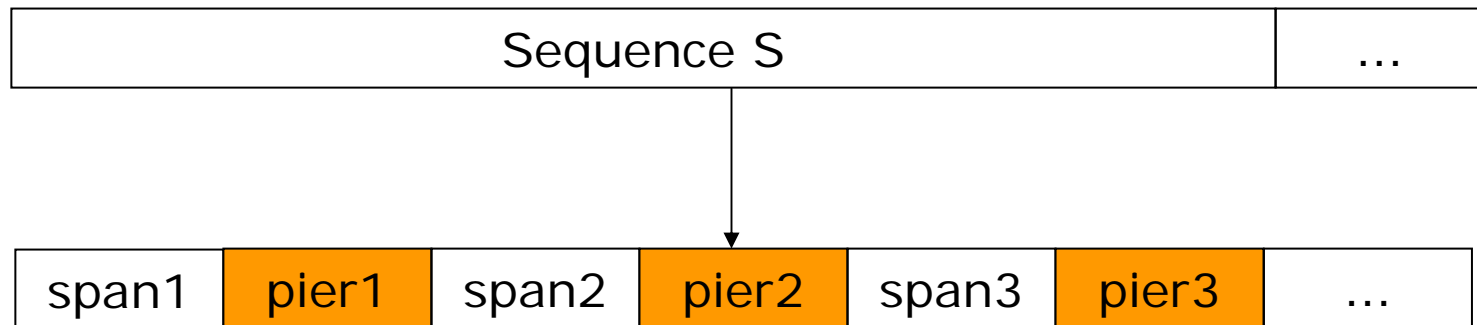
Piers: An Efficient Model for
Similarity Search in DNA
Sequence Databases

Outlines

- Notations
- Pier model and sensitivity analysis
- Hash-based pier model
- Sequence similarity search using hash-based pier model
- Experimental studies
- Conclusion

Notations

- **Pier:** A pier is defined as a pair $\langle p, \text{pos} \rangle$. Pier sequence p is a segment extracted from DNA sequence of length l_p .
- **Span:** The span is the segment between two adjacent piers.



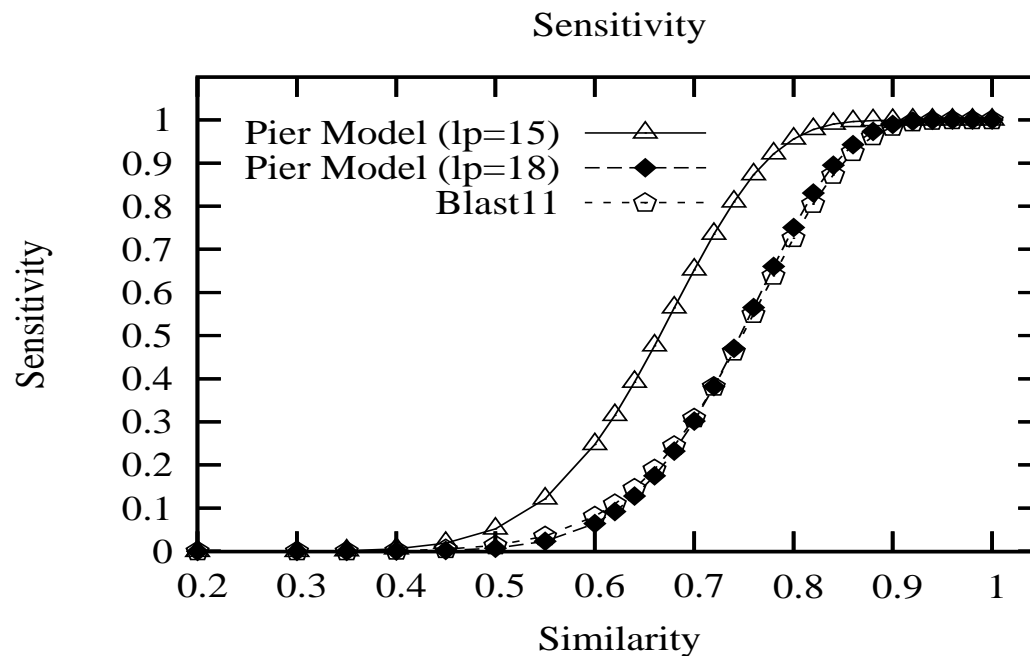
The Pier Model

- Assumption
 - Users are only interested in high similarity regions that are of length greater than a minimum length l_{\min} .
- Pier Extraction Principle
 - At least k piers should be contained in any subsequence with length no less than the threshold of minimum length l_{\min} . This means that the following formula must hold: $((k+1)l_p + kl_s) \leq l_{\min}$.

- Piers are first extracted from data sequences.
- The existing similarity search methods are applied on the set of piers instead of the whole sequence database.

Theoretical Sensitivity Analysis

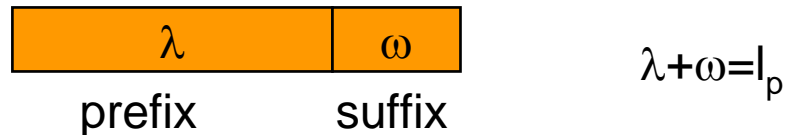
- Sensitivity is the probability that two sequences with the same length and a certain level of similarity can be regarded as a similar pair, or a hit.



L=66, (1) lp=15, ls=7, $\epsilon=3$; (2) lp=18, ls=9, $\epsilon=3$.

The Hash-based Pier Model

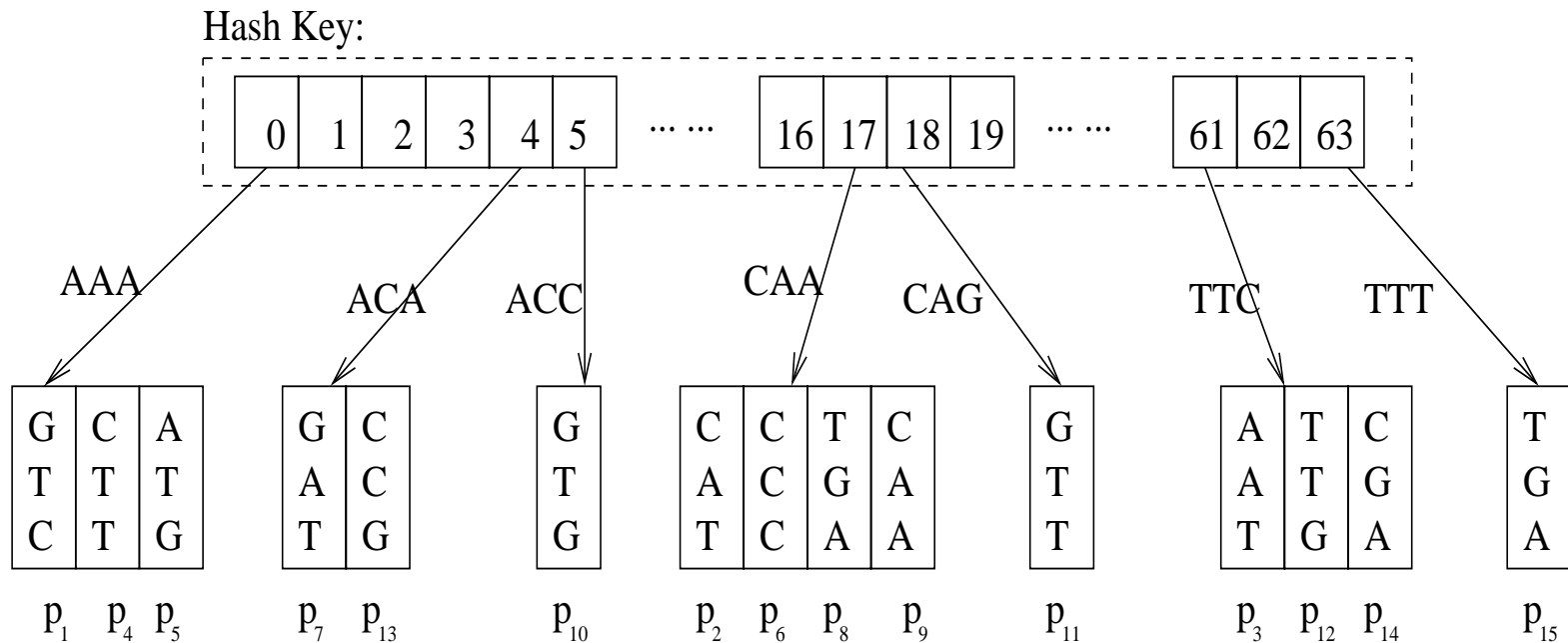
- The pier segment is divided into two parts:



- The prefix is encoded to a 2λ bit integer:
 - A- \rightarrow 00, C- \rightarrow 01, G- \rightarrow 10, T- \rightarrow 11;
- The size of hash table: $2^{2\lambda} = 4^\lambda$

The Hash-based Pier Model (Cont.)

- $l_p = 6, \lambda = 3, \omega = 3;$
- Piers: $P = \{p_1, \dots, p_{15}\};$
- $4^\lambda = 64$ buckets in hash table;
- In experiments, $\lambda = 10, 4^\lambda = 10^6 = 1M.$



Collision Handling

- Use global penalty matrix (GPM) to handle the collision list of a hash table
 - $4^{2\omega}$ global penalty matrix, $\omega = l_p - \lambda$.
 - Compute edit distance for each pair of $\langle i, j \rangle$ where $0 \leq i, j < 4^\omega$, and store the value in cell $\langle i, j \rangle$ in GPM.

Query Processing

- Generate the query pattern with size of l_p from Q .
- Search for pier candidates among the hashed piers.
- Post-process the candidates to concatenate adjacent candidates to form final alignments with a high alignment score.

Query Processing -- Neighborhood Enumeration

- Let S and Q be two sequences of length λ .
- If $\text{edit}(S, Q) \leq 2$, then one of the following cases is true:
 - $\text{edit}(S, Q) = 0$, i.e. the two sequences are exactly the same;
 - $\text{edit}(S, Q) = 1$, i.e. one replacement operation is needed in S to transform S to Q ;
 - $\text{edit}(S, Q) = 2$, there are three subcases
 - two replacement operations in S are needed to transform S to Q ;
 - one insertion and one deletion in S with order are needed to transform S to Q ;
 - one deletion and one insertion in S with order are required to transform S to Q .

Query Processing -- Sequence Similarity Search

- Enumerate an encoded neighbor **ngbr** of the hash key of query pattern q ;
- Retrieve the piers in the bucket of the hash structure **HTable[ngbr]**;
- Use the global penalty matrix (GPM) to see if the retrieved piers are similar to the query pattern q .

Space Complexity

- hash table head: $O(4^\lambda)$
- the bucket of the table: each pier will contribute space $O(\omega)$. Thus, the total size of the buckets will require space $O(\omega|P|)$.
- global penalty matrix: $O(4^{2\omega})$
- total space complexity for the hash structure will be $O(4^\lambda + \omega|P| + 4^{2\omega})$.

Datasets

Database	Size
<i>ecoli.nt</i>	4.68M
<i>yeast</i>	12.3M
<i>month.gss</i>	286.2M
<i>patnt</i>	702.1M
<i>other_genome</i>	1.06G
<i>other_genome + patnt</i>	1.76G

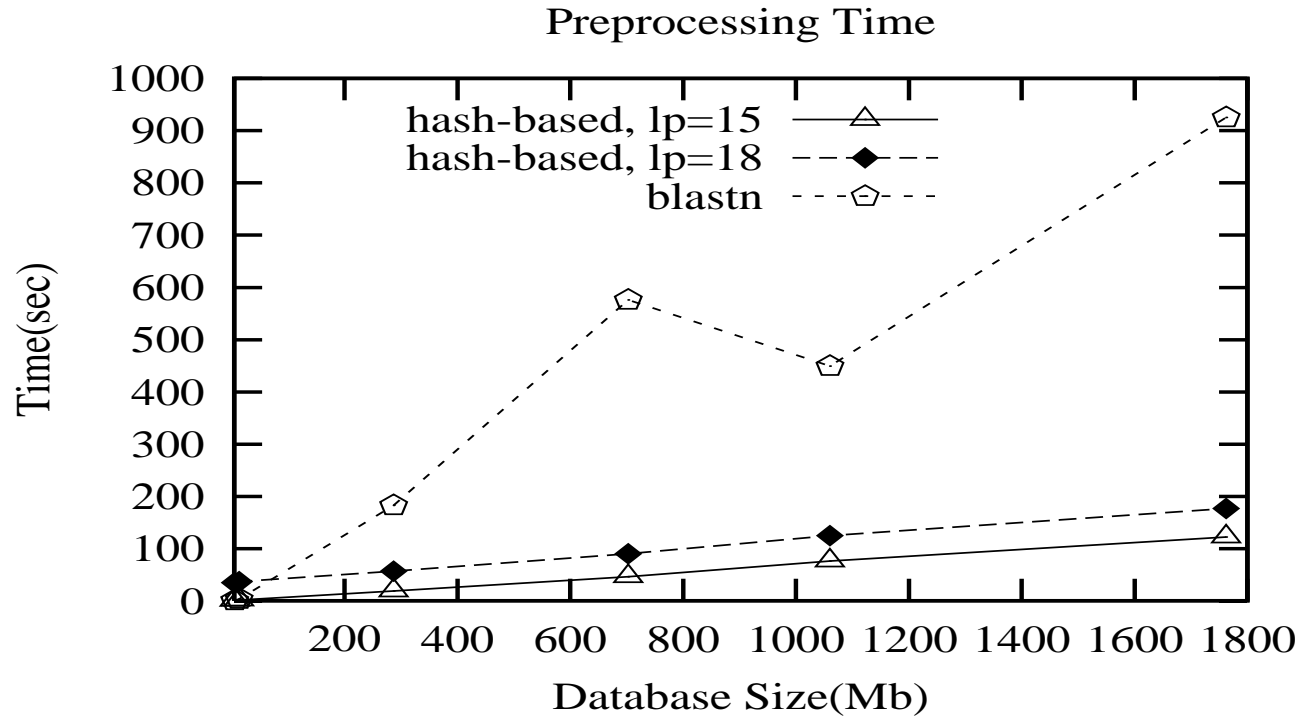
Table 3.3: The DNA Sequence Databases

Parameter Settings

Parameter	Values in Case I	Values in Case II
l_p	15	18
l_s	7	9
λ	10	12
θ	3	3

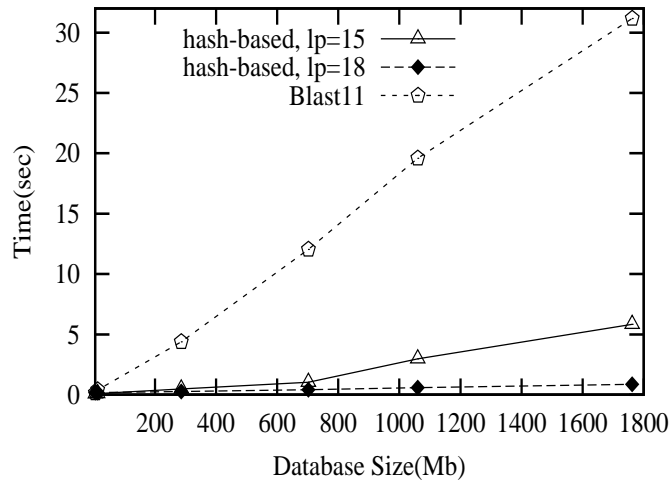
Table 3.4: The Parameters in the Experiment

Experimental Results

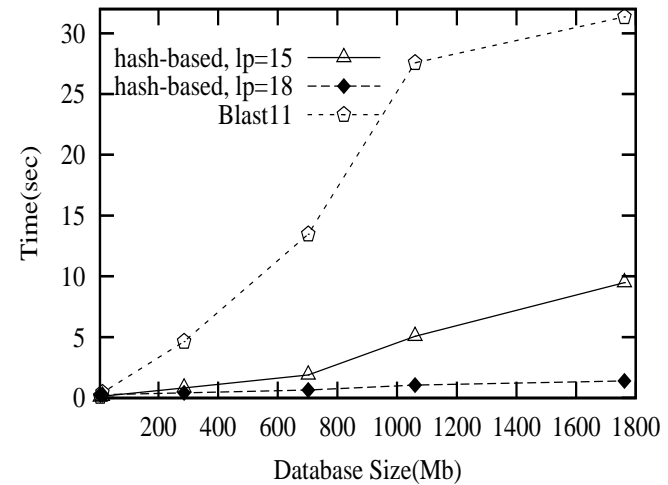


Experimental Results (Cont.)

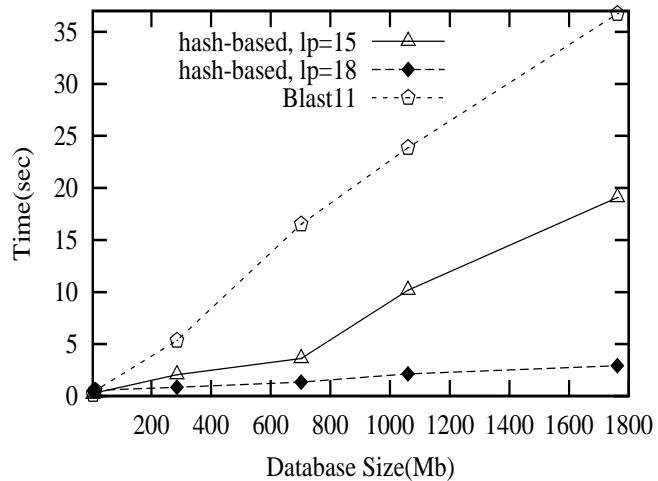
Query length=300



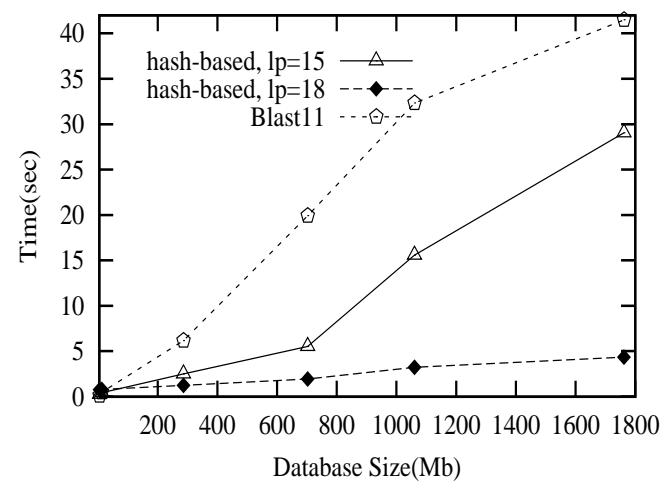
Query length=500



Query length=1000

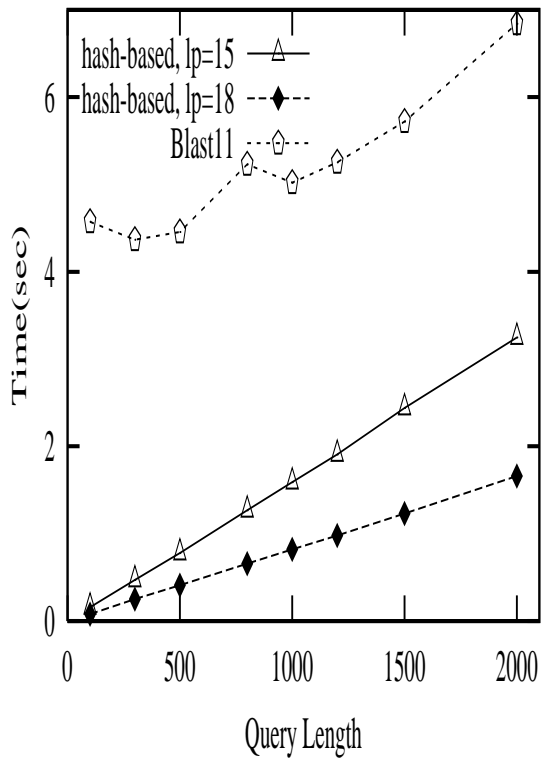


Query Length=1500

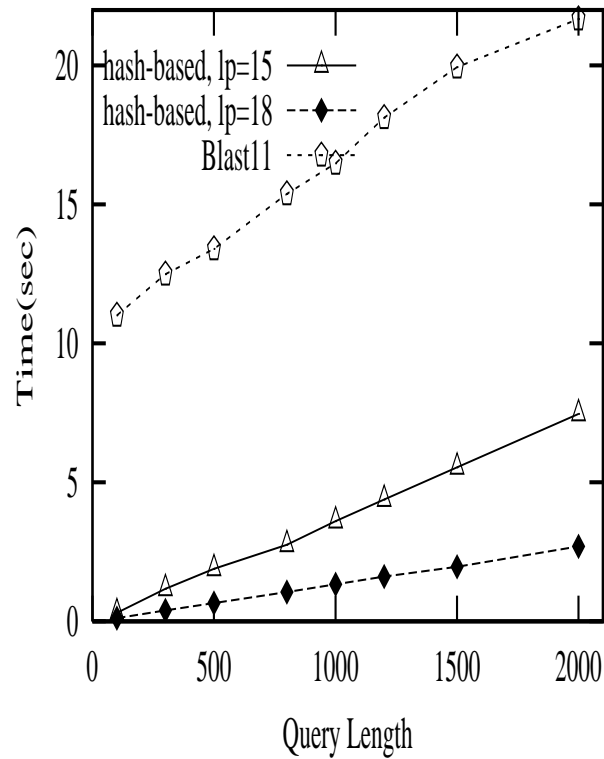


Experimental Results (Cont.)

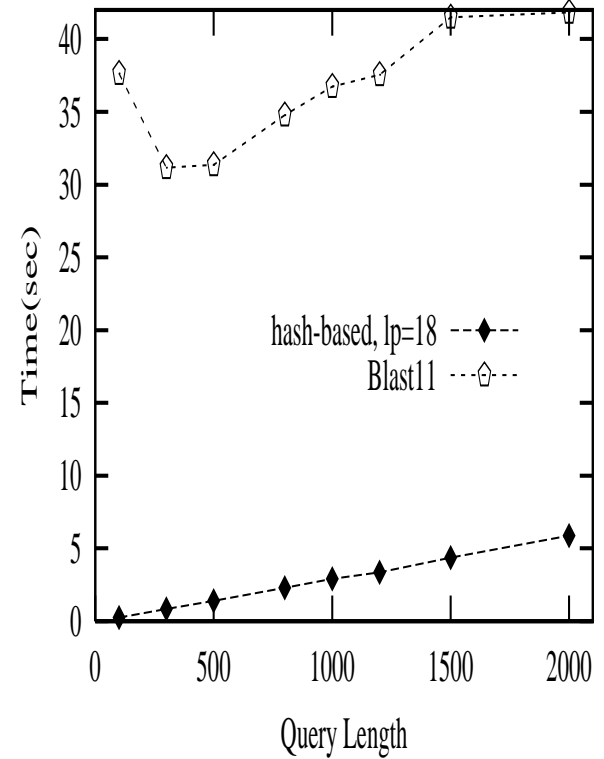
DB:month.gss



DB:Patnt



DB:Other_genome+Patnt



Summary

- ❑ Developed the hash-based pier model;
- ❑ Demonstrated its search efficiency and sensitivity;
- ❑ Also proposed a method -- the GPM-based method -- to further improve search efficiency by reducing the computation cost of candidate verification;
- ❑ The proposed method is faster, yet requiring smaller memory and space.

Solution 2



Indexing DNA Sequences Using
q-grams

Notations

- q-gram of Sequence
 - Given a sequence S , its q-grams are obtained by sliding a window of length q over the characters of S . For a sequence S , there are $|S|-q+1$ q-grams.
- q-gram Cluster
- q-gram Signature
- c-signature

The q-gram Cluster

- DNA: {A, C, G, T}
- Number of q-grams in DNA sequence: 4^q

- $q=2$
- q-grams: {AA, AC, AG, AT, CA, CC, CG, CT, GA, GC, GG, GT, TA, TC, TG, TT}.

We simply cluster the m continuous q-grams together into a qCluster;

There are $\lambda(=8)$ q-gram Clusters for $q=2$, $m=2$;

$qc_1: \{AA, AC\}$	$qc_5: \{GA, GC\}$
$qc_2: \{AG, AT\}$	$qc_6: \{GG, GT\}$
$qc_3: \{CA, CC\}$	$qc_7: \{TA, TC\}$
$qc_4: \{CG, CT\}$	$qc_8: \{TG, TT\}$

Signatures for DNA Sequence

•P: "ACGGTACT"
•q=2

q-grams: {AC, CG, GG, GT,
TA, AC, CT}

q-gram signature: (01 00 00
11 00 11 10 00) with 16 (4^2)
dimensions

The c-signatures for sequence P:

(01 00 00 11 00 11 10 00)

c=2: (1 0 0 2 0 2 1 0)

(010 000 110 011 100 0)

c=3: (1 0 2 2 1 0)

Lemma: Filter based on q-grams (Jokinen and Ukkonen, 1991)

- $Q[1:w]$
- $S[j-w+1:j]$
- at most ε edit or hamming distance
- q-grams in Q and S

There are at least $w+1 - (\varepsilon+1)q$ of the q-grams in $Q[1:w]$ which occur in the substring $S[j-w+1:j]$.

There are at most εq q-grams in $Q[1:w]$ which do not occur in $S[j-w+1:j]$, and vice versa.

The number of different q-grams between $Q[1:w]$ and $S[j-w+1:j]$ is at most $2\varepsilon q$.

Lemma: Filter Based on c-signatures

- DNA Sequences: S, P
- at most ε edit or hamming distance
- q-gram signatures ($c=1$): $\text{sig}^1(S) = (a_0, a_1, \dots, a_{n-1}), \text{sig}^1(P) = (b_0, b_1, \dots, b_{n-1}), n=4^q$
- c-signatures ($c>1$): $\text{sig}^c(S) = (u_0, u_1, \dots, u_{k-1}), \text{sig}^c(P) = (v_0, v_1, \dots, v_{k-1}), k=4^q/c$

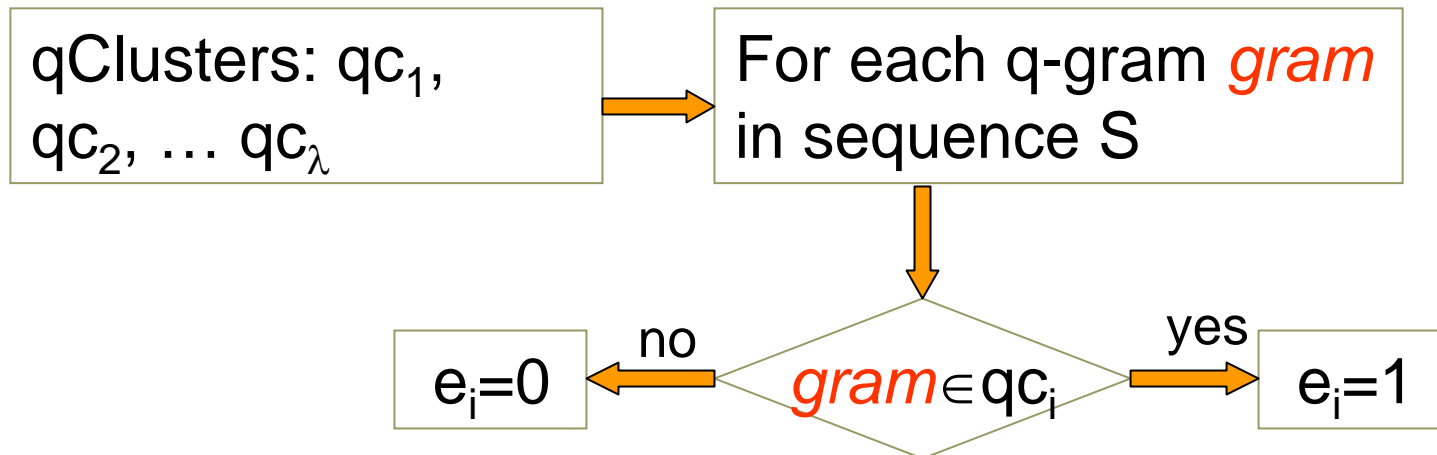
$$\sum_{i=0}^{k-1} |u_i - v_i| \leq \sum_{j=0}^{n-1} |a_j - b_j| \leq 2\varepsilon q$$

A two-level Indexing Scheme for DNA Sequences

- Level 1: **Hash Table** built on qClusters of DNA sequences;
- Level 2: the **c-trees** built on c-signatures of DNA sequences;

Level 1: The Hash Table

- Encode the DNA segment S into a λ -bit integer.



- Each sequence S can be transformed into a λ -dimensional bitmap (e_1, \dots, e_λ) .
- Encoding function **coding(S)**: $\text{coding}(S) = \sum_{i=1}^{\lambda} 2^{i-1} e_i$
- Note that λ is set 22 for $q=4$ in the experimental studies (hash table size is $2^\lambda \approx 4\text{MB}$).

Level 2: The c-trees Construction

□ The five DNA segments

- $s_0 = \text{ACGGT}$,
- $s_1 = \text{CTTAG}$,
- $s_2 = \text{ACGTT}$,
- $s_3 = \text{TAAGC}$,
- $s_4 = \text{GACGT}$.

□ $q=2, c=2$

• The c-signatures are:

$$\text{sig}^2(s_0) = (\underline{1001} \ \underline{0200}),$$

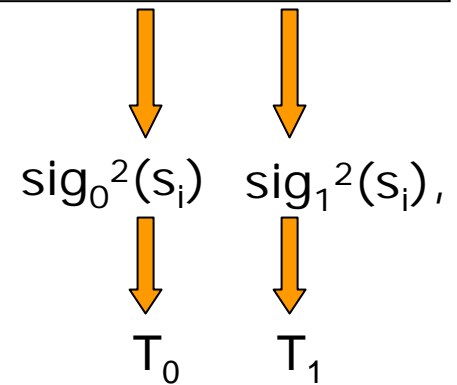
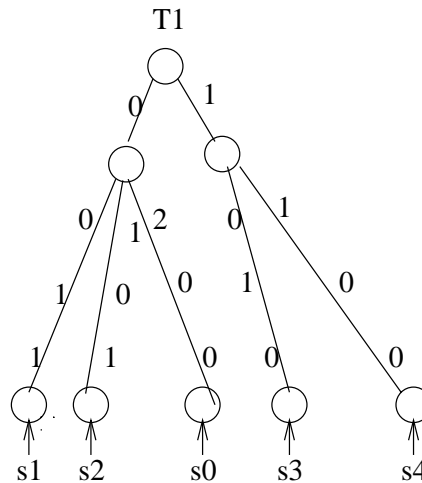
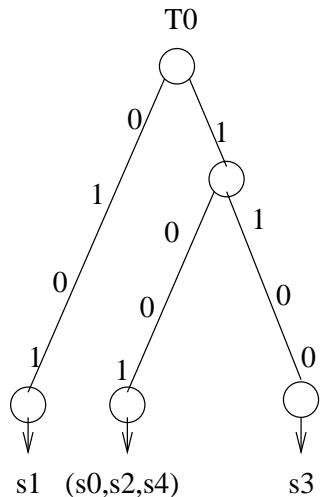
$$\text{sig}^2(s_1) = (\underline{0101} \ \underline{0011}),$$

$$\text{sig}^2(s_2) = (\underline{1001} \ \underline{0101}),$$

$$\text{sig}^2(s_3) = (\underline{1100} \ \underline{1010}),$$

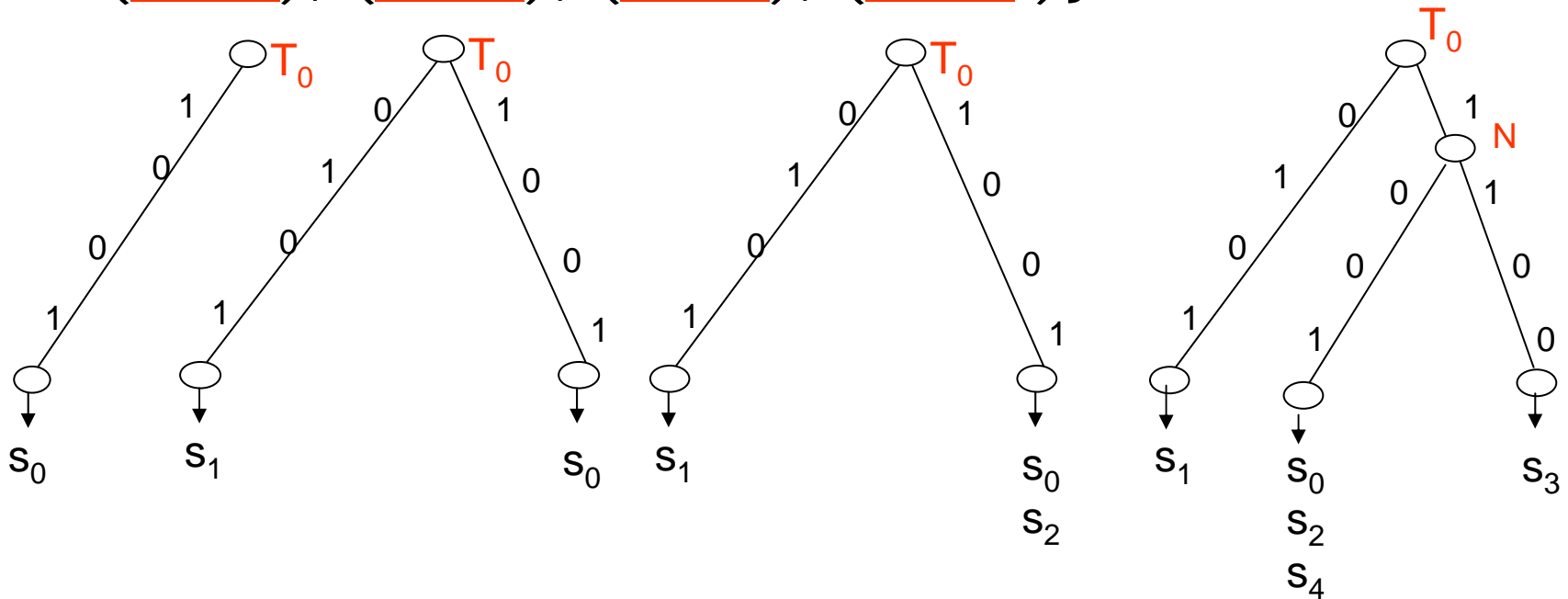
$$\text{sig}^2(s_4) = (\underline{1001} \ \underline{1100}).$$

If tree depth h is 4, there are $4^q/(c \times h) = 2$ trees, T_0, T_1 .



The c-trees Construction (Cont.)

- For T_0 , the c-signature strings are $\{(\underline{1001}), (\underline{0101}), (\underline{1001}), (\underline{1100}), (\underline{1001})\}$.



Query Processing

- Sequence Database:
 - The disjoint segments with the length ω are generated from the sequences.
 - Two-level index: a hash table HT and the c-trees are built on the DNA segments.
- Query Sequence:
 - Query sequence Q is decomposed into $|Q| - \omega + 1$ sliding query patterns $\{q_1, \dots, q_{|Q| - \omega + 1}\}$.
 - The two-level index is used to identify potential candidates by pruning data segments that are far away from the query sequence.
 - The dynamic programming is conducted to obtain the final alignments with high alignment score between the candidates and query sequence.

The First Level Filter: Hash Table Based Similarity Search

1. A hash table HT built on data segments
2. Query pattern q_i is also encoded into a hash key h_i .

e_{ngbr} : Neighbors of h_i can be found by enumerating all substrings that have edit distance $\leq d$ from q_i .

For an encoded neighbor e_{ngbr} of q_i , the segments in $HT[e_{ngbr}]$ will be retrieved as candidates, and stored in candidate set C_{ht} .

The Second Level Filter: The c-trees Based Similarity Search

Inputs:

- $\{T_0, \dots, T_{\delta-1}\}$: the c-trees built on DNA data segments;
- $sig^c(q)$: the c-signature of query q ;
- Divide $sig^c(q)$ into δ strings which are $sig^c_i(q)$ $0 \leq i < \delta$

Method:

1. For each leaf node $INode$ in tree T_0 , the distance $w_0[INode]$ between the path label of $INode$ and $sig^c_0(q)$ is computed.
2. Initialize candidate set $C = E_0[INode] \cap C_{ht}$, where $w_0[INode] \leq \text{threshold}(=2\epsilon q)$.
3. For the trees $\{T_1, \dots, T_{\delta-1}\}$, candidates C will be pruned on partial distance gradually.

Output:

1. Candidate C
2. The dynamic programming is conducted to obtain the final alignments with high alignment score.

Space Complexity

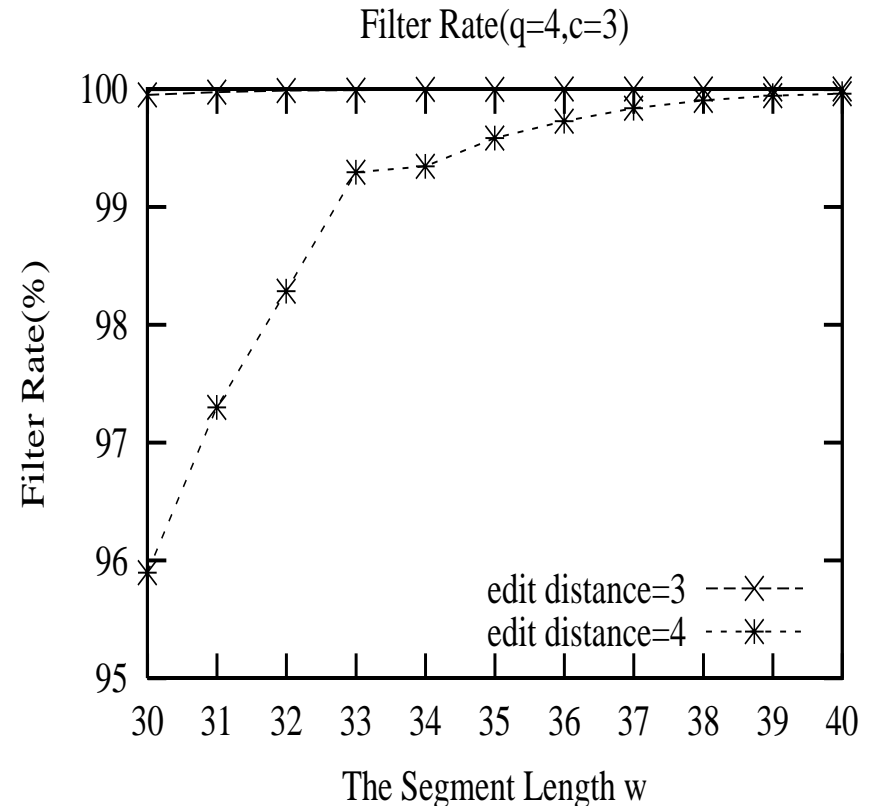
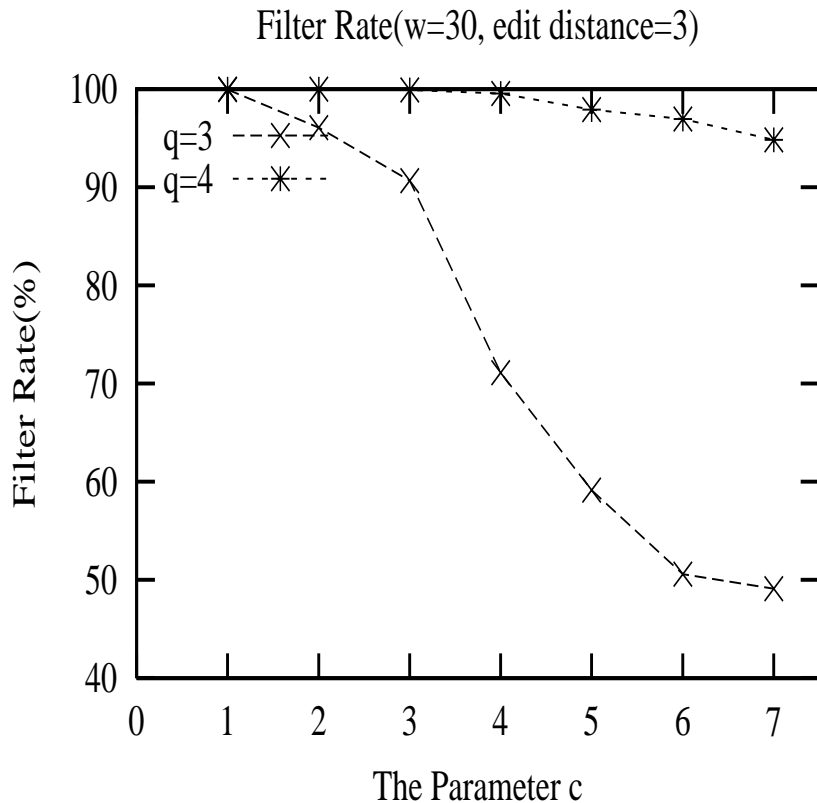
- Hash table: the total space complexity for the hash structure will be $O(2^\lambda + |D|/\omega)$.
- The c-trees: the space complexity for the c-trees can be divided into two portions:
 - the c-trees themselves: the storage required for the edge labels is bounded by $O((c+1)^{4^q/(\delta c)} \log(c+1))$ for each tree.
 - space occupied by $E_0[*]$ and the links:
 - $E_0[*]$ must be stored for the first tree; thus they require $O(|D|/\omega)$ space.
 - we also need to maintain the links for the other trees. The space required by links highly depends on the data distribution. Note there are lots of zeros in c-signatures, thus a lot of links will point to a dummy leaf (by dummy we mean that the path label is 0). So we can compress those links.

Datasets and Experiments

- The five DNA sequence datasets used in the experiments are:
 - other_genomic(1.06GB),
 - Patnt(702.1MB),
 - month.gss(286.2MB)
 - yeast.nt(12.3MB)
 - ecoli.nt(4.68MB).
- Effectiveness
- Sensitivity
- Efficiency
- Latest version of BLAST(NCBI BLAST2), seed length=11.

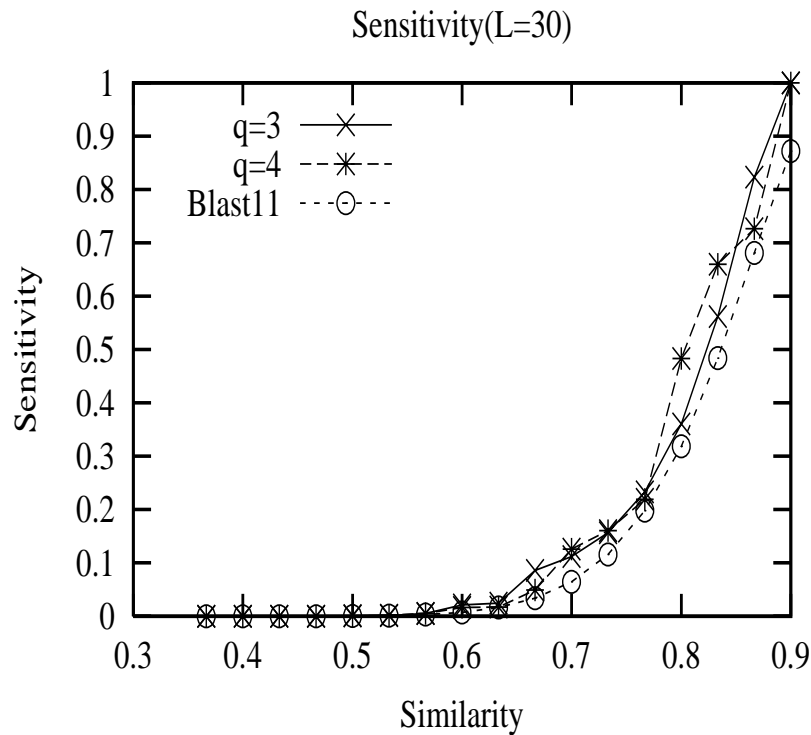
The Effectiveness Analysis

- The filter rate is defined as the ratio of the total number of hits found to the total number of segments in data sequence.

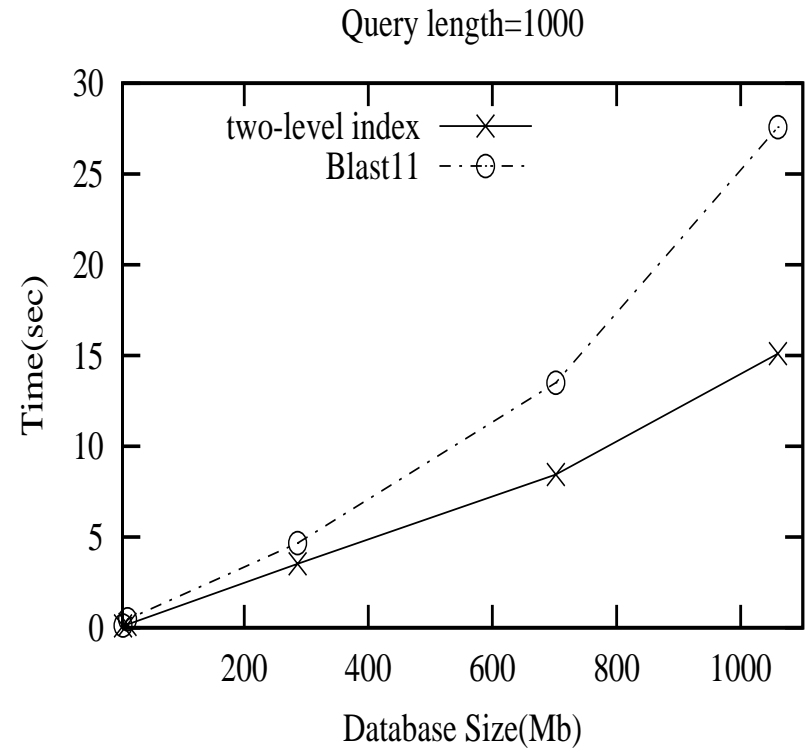
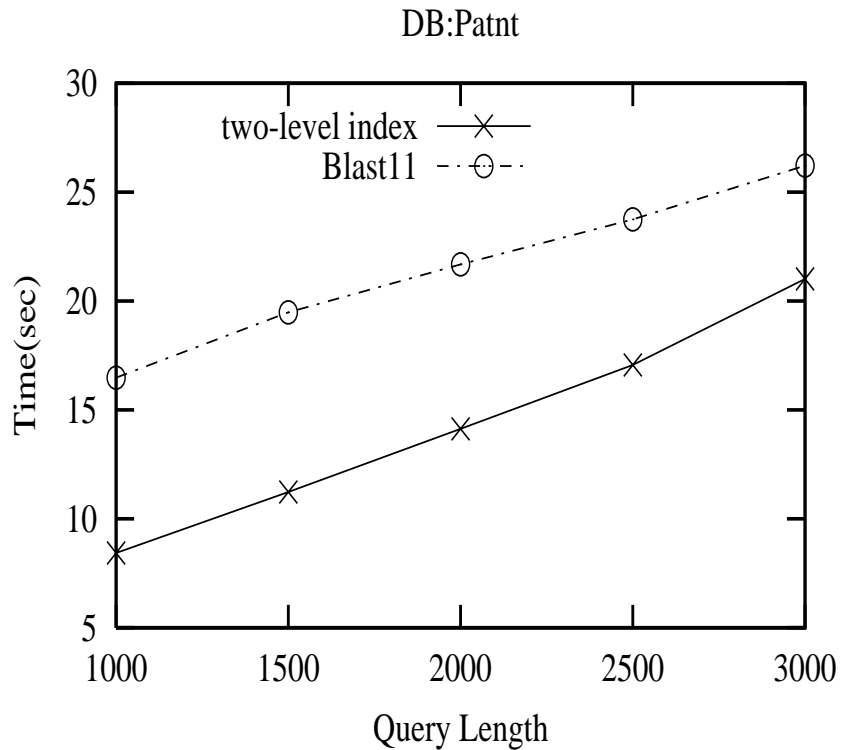


The Sensitivity Analysis

- The sensitivity is the probability that the two sequences S and P with length L and similarity sim can be regarded as similar sequences.



The Efficiency Analysis



Summary

- A novel two-level index structure based on q-grams is devised for DNA sequences which can support efficient similarity search in DNA sequence database.
- The filter principle with respect to the index structure is presented.
- The experimental results show that our method can efficiently detect the regions in DNA sequence database which are similar to the query sequence with high sensitivity.

String Join Using Precedence Count Matrix



Problem II

Outline

- Problem Definition
- Approximating Edit Distance Using Precedence Count Matrix (PCM)
 - Definition and property of PCM
 - Algorithm of estimating edit distance using PCM
- Approximate DNA Sequence Join Algorithm
- Performance Study
- Conclusion

Problem Description

- Sequence join is one of the most frequently used and expensive operations that combines data from two datasets with similar string values on the join attribute.
- In our work, two sequences are joinable if the prefix of one sequence is similar to the suffix of another.

Definition of PCM

- Definition: Precedence Count Matrix (PCM)
 - Σ = the set of alphabet;
 - Q = a sequence formed from Σ ;
 - PCM_Q is a $|\Sigma| * |\Sigma|$ matrix where $PCM_Q[a,b]$, $a \in \Sigma$, $b \in \Sigma$, is the number of unique occurrences of a preceding b (not necessary consecutive) in the sequence Q .

The PCM for the DNA sequence
 $Q = \text{"AAATGTTCCAC TTCGGGCC"} ,$
where $\Sigma = \{A, C, G, T\}$,

Note: There are 7 'C' following the first 3 'A', and 4 'C' following the fourth 'A', so $PCM_Q(A, C) = 3 * 7 + 1 * 4 = 25$.

	A	C	G	T
A	6	25	15	17
C	3	21	15	8
G	1	13	6	4
T	3	27	16	10

PCM_Q

Algorithm of Estimating Edit Distance Using PCM

Input: PCMs of two sequences Q and R

Output: Lower bound of edit distance
between Q and R

Method:

step1. Adjust Diagonal Elements

step2. Compute Maximum Impact

step3. Adjust Non-Diagonal Elements

- Theorem: The lower bound of edit distance between two DNA sequence Q and R can be computed in $O(|\Sigma|^2 \log |\Sigma|)$ using PCM_Q and PCM_R.
- It can be proved by analyzing the 3 steps of our algorithm.
- It can be used in approximating the edit distance of two DNA sequences efficiently since $|\Sigma|=4$ for DNA sequences.

PCM Based Method for DNA Sequence Join

- Transformation
 - For both suffix dataset and prefix dataset, transform sequences into its corresponding PCMatrix
- Filtering
 - Use the filtering techniques based on PCM to obtain the candidates pairs for sequence joins
 - Distance Filtering
 - Use the lower bound of edit distance based on the transformed PCMs as the efficient filtering without false dismissal;
 - If $PCMEdit(S,P) > e$, then $edit(S,P) > e$;
 - Length Filtering
 - If $|\text{len}(S) - \text{len}(P)| > e$, then $edit(S,P) > e$
- Verification
 - Obtain the final results pairs of DNA sequence joins by using dynamic programming

Performance Study

□ Dataset

- 1000 DNA sequences with the length varying between 200 and 300 from a complete sequence in Ecoli sequence database

□ Methods:

- Compare our scheme against q-grams(qgram) method and frequency vector(FV) method
- Also compare against two integrated strategies: PCM+qgram, FV+qgram

Performance Study (Cont.)

e	PCM	PCM+qgram	FV	FV+qgram	qgram
1	99.9985%	100%	99.81%	100%	100%
2	99.42%	99.998%	96.66%	99.998%	99.997%
3	92.47%	99.986%	84.98%	99.98%	99.98%
4	75.17%	99.918%	66.39%	99.91%	99.90%
5	54.42%	99.387%	47.25%	99.36%	99.28%

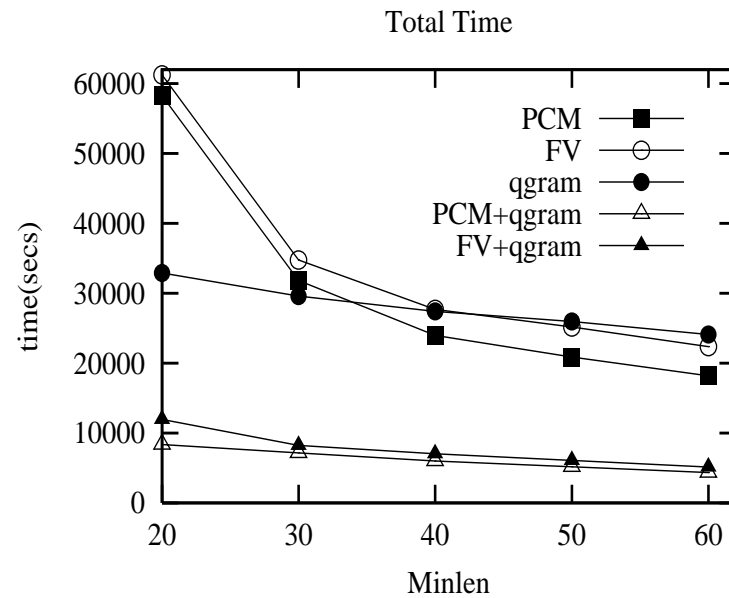
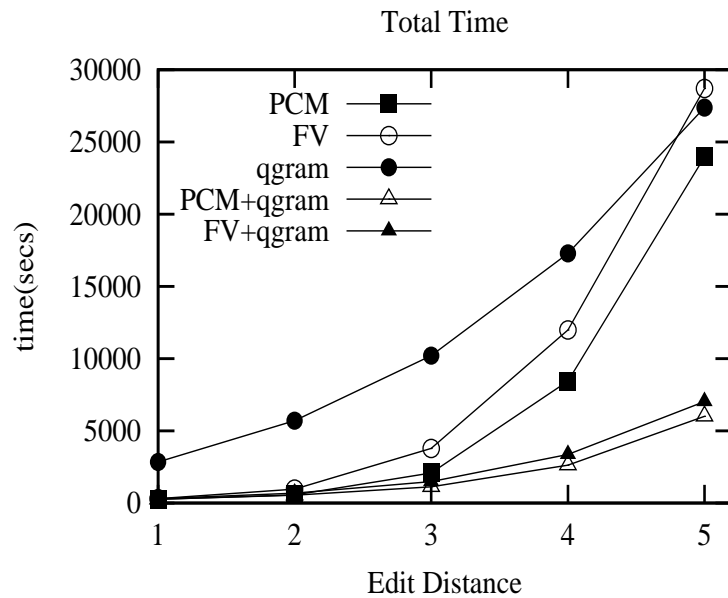
Filtering Rate for Minlen=40 (Filtering Rate = number of the candidates filtered / all possible combinations)

<i>l</i>	PCM	PCM+qgram	FV	FV+qgram	qgram
20	20.50%	32.09%	14.99%	27.91%	80.97%
30	39.17%	92.17%	32.27%	91.69%	90.73%
40	54.42%	99.39%	47.25%	99.36%	99.28%
50	64.74%	99.88%	58.36%	99.87%	99.87%
60	71.92%	99.96%	66.46%	99.96%	99.96%

Filtering Rate for e=5

Performance Study (Cont.)

Effect of edit distance e and Minlen



Summary

- Propose a filter-and-refine string join algorithm for genomic applications;
- Employ an efficient algorithm to remove the remaining false candidates in refinement phase;
- Use PCM to compute the edit distance between two DNA sequences efficiently;
- The proposed sequence join algorithm outperforms known techniques in our experiments.

The q-gram Based Protein Subcellular Localization Prediction



Problem III

Problem Statement

- Given an unlabeled protein sequence S , and a known subcellular localization L , we want to determine if S locates in L .
 - Given the positive protein sequence dataset $P = \{p_0, \dots, p_m\}$ and negative protein sequence dataset $N = \{n_0, \dots, n_n\}$
 - Extract the features in protein sequences
 - Develop a classifier on these features to distinguish the sequence p_i in P from n_j in N .

Methods

- Proposed and generated the q-gram based features to capture the information in protein sequences
 - q-gram frequency vectors
 - q-gram similarity vectors
 - q-gram TF.IDF vectors
- The q-gram based feature vectors are used to train the support vector machine (SVMs) to predict the protein subcellular localization

Prediction Results for All Subcellular Localizations

localization	method	q	precision	recall	accuracy	specificity	MCC
outer	Frequency	1	88.22%	84.36%	92.71%	95.81%	0.813284
outer	DWT	1	95.53%	77.44%	92.92%	98.67%	0.816776
outer	Similarity	3	91.37%	81.79%	92.99%	97.14%	0.818453
outer	TF.IDF	3	93.47%	75.64%	91.95%	98%	0.791047
cyto	Frequency	1	81.10%	65.82%	90.50%	96.39%	0.675487
cyto	DWT	1	89.03%	56.36%	90.28%	98.28%	0.657381
cyto	Similarity	3	81.83%	74.91%	92.01%	96.05%	0.734141
cyto	TF.IDF	3	82.69%	56.36%	89.38%	97.17%	0.625097
cm	Frequency	1	98.09%	81.64%	95.76%	99.56%	0.870624
cm	DWT	1	100.00%	73.12%	94.31%	100.00%	0.825489
cm	Similarity	3	95.25%	85.57%	96.04%	98.85%	0.878768
cm	TF.IDF	3	100.00%	77.05%	95.14%	100.00%	0.851551
extra	Frequency	1	65.71%	67.37%	91.11%	94.72%	0.613877
extra	DWT	1	96.27%	56.32%	93.96%	99.68%	0.708682
extra	Similarity	3	86.77%	61.05%	93.61%	98.56%	0.694617
extra	TF.IDF	3	89.92%	51.05%	92.85%	99.20%	0.641817
peri	Frequency	1	74.17%	67.27%	88.96%	94.08%	0.637714
peri	DWT	1	89.37%	52.36%	89.58%	98.37%	0.631646
peri	Similarity	3	74.30%	62.91%	88.68%	94.76%	0.615731
peri	TF.IDF	3	85.82%	50.91%	89.03%	98.03%	0.606374

Table 6.7: Results for different methods based on q -grams

Thank you!