Development of Biological Sequence Approximate Matching Algorithms through the Optimization of Reuse-Based Design

Hai Long[1], Haihe Shi[1,*], Linhui Zhong[1], Haipeng Shi[1], Songqing Xu[2], Jun Sun[2], Yong Qiu[2], Changsheng Hu[2]

[1]School of Computer Information Engineering, Jiangxi Normal University, Nanchang 330022, China
[2]Jiangxi Cloudnet Technology Co., Ltd., Nanchang 330036, China
* Corresponding Author

Abstract: Biological sequence approximate matching algorithms (BSAM) are often used to compare DNA or protein sequences of different species, and to study the relationship between sequences and the function of sequences. The existing research mainly focuses on the step optimization or parallel implementation of such algorithms to improve efficiency, but the lack of algorithmic framework research with high domain level abstraction leads to the complexity and incomprehension of the current BSAM algorithm, and it is difficult for researchers to choose algorithms that meet individual needs. By deeply analyzing the BSAM algorithm domain, the feature-oriented domain modeling method and the idea of software reuse are used to extract the common features and separate the heterogeneous features of the BSAM domain, so as to design reusable components, and further combine the BSAM problem solving process to design the BSAM domain feature model and component interaction model. It is implemented using a highly trusted PAR platform. By assembling a new variant of the algorithm Misq, we demonstrate the optimization effect of reusable components on algorithm design.

Keywords: sequence approximate matching, domain modeling, feature model, interaction model, PAR platform

1. Introduction

The genetic sequence and protein sequence of an organism preserve various information about the organism. In terms of species evolution, biologists can compare the DNA or protein sequences of different species, study the evolutionary relationship between them through sequence matching to determine the common ancestor between species, and infer the evolutionary history of different species; In terms of gene prediction, a new DNA or protein sequence can be compared to a known sequence library to determine its function. If the two sequences are very similar, then they may have similar functions. Therefore, sequence analysis can help us understand the relationship between these sequences, helping biologists to study problems such as species evolution, based on function and biological processes. The research basis of gene sequence alignment is also biological sequence approximation matching technology. Since the first generation sequencing technology was proposed in 1977, to the now widely used second-generation sequencing technology, and then to the third-generation sequencing technology that has made breakthroughs over the years, biological genetic data has been growing, and how to efficiently store, access and query these massive genetic data is an urgent problem[1].

Biological sequence approximate matching algorithms (BSAM) is one of the important means to deal with the above problems. For the BSAM method, FASTA [2] and BLAST [3] are the two main heuristics for similar queries based on the q-gram[4] indexing mechanism, and although they are very efficient, they may not find the optimal solution and may require a large amount of memory[5]. The methods for finding optimal solutions in BSAM can be divided into four main methods [6]: dynamic programming matrices, finite automata, bit parallelism, and filters. The most basic method is the Dynamic Programming Matrix [7] proposed by P. Sellers in 1980 (Dynamic Programming Matrix, DPM). It calculates all the values of a two-dimensional array of size ($m*n$) to determine which regions in the database sequence match the query sequence, where m and n are the length of the query sequence and the database sequence, respectively, and their time complexity is O(mn) at worst. However, for biological sequences, a sequence can be millions in length and numerous, and it is too inefficient to use this method directly, and most of these methods are online sequence approximation matching algorithms, which are not advantageous over offline sequence approximation matching algorithms with index structures when performing multiple matches for the same set.

The offline biological sequence approximation matching algorithm is currently a widely used and efficient BSAM algorithm. Its threshold-based BSAM algorithm is very active[4,8-11] and has good results in the field of approximate matching. The q-gram algorithm proposed in 1991[4] is a threshold-based sequence approximation matching algorithm, which mainly divides the string of equal length to obtain the gram features of the sequence, then establishes an inverted index of the gram features, uses count filtering as the similarity comparison function to obtain the intermediate candidate sequence, and then

verifies the online sequence approximation matching algorithm of dynamic programming to obtain the final result. The sequence division method of this method is simple and does not need to be supported by word segmentation tools, but the established index is large. The SSjoin algorithm [8] proposed in 2006 is an algorithm based on prefix filtering method for approximate sequence matching, and the use of prefixes greatly reduces the space for indexing. The Ed-join [9] algorithm proposed in 2008 uses non-matching as the filtering standard, starting from the aspects of count and content. In 2008, Chen Li [10,11] et al. proposed a variable-length feature filtering method called vgram, which first needs to count the frequency of the occurrence of various lengths of gram in the collection, and then select gram through frequency information, and use a heuristic rule to extract higher quality grams, which reduces the size of the inverted index linked list, and also leads to length filtering and position filtering, the former is a string-based filtering method. The latter is a feature-based filtering method. In 2013, Wang et al. proposed a sequence approximation matching method called vchunk [12] based on variable-length chunk features, which uses chunks to limit the tail part of the string, thereby dividing the string into variable-length non-intersecting chunks. In addition, there are some algorithms based on the BWT algorithm [13,14], the suffix tree[15,16] and the suffix array[17] that have also achieved good results.

There are many algorithms in the field of BSAM, but for the ever-increasing biological database, the BSAM algorithm has the lack of filtering criteria, each filtering criterion is limited to each algorithm and cannot be expanded, the filter area selection is too complex or simple, and the division of features and filtering means require personalization. At present, the field focuses on optimizing certain steps of algorithms [18] or parallelizing the implementation of algorithms[19] to improve the efficiency of algorithms, which cannot deal well with these problems. At the same time, for the BSAM algorithm, the researchers are only concerned with the algorithm that looks wonderful in theory, that is, it has good time complexity, while the developer only pursues the fastest possible algorithm in practical application[20], which makes it difficult to combine theoretical research and practical application. Therefore, it is extremely important to study the sequence approximation matching algorithm from the domain abstraction level. By extracting the common features and heterogeneous features in the BSAM field, combining the algorithm process, establishing a feature model, designing interaction models and algorithm components, constructing the algorithm theoretical framework, and then assembling the corresponding BSAM algorithm, optimizing the implementation process of the algorithm, integrating various heterogeneous features, realizing the personalized requirements of the algorithm, reducing the redundancy of algorithms in the development and use of such fields, and improving the reliability of the algorithm.

Next, Section 2 introduces the related techniques used in the paper, namely the feature-oriented domain modeling method and the PAR (Partion And Recur) method. Section 3 introduces the relevant concepts in the BSAM field, deeply analyzes the BSAM field, uses the feature-oriented domain modeling method FODM (Feature-Oriented Domain Model)[21] to analyze and design the BSAM algorithm, uses reusable design ideas to separate common features and heterogeneous features, combines the algorithm process, establishes a feature model in the field, and designs an interaction model. In this section, the abstract generic algorithm components in the BSAM field are designed, and the abstract components are realized using the new high-reliability software development platform PAR[22-27] proposed by Xue Jinyun, and the components are assembled. A new variant algorithm that meets the personalized needs of the algorithm is generated through experimental assembly, and the generated algorithm is experimentally analyzed by using biological sequences. Finally, the full text is summarized.

2. Related Theories and Methods

2.1. Domain Engineering

Domain engineering [28] is the main technology for the production of reusable software assets, and it is the process of establishing basic capabilities and necessary foundations for the application engineering of a group of similar or similar systems [21]. Domain engineering helps to produce components with high reusability. Use domain engineering to analyze the same statutes and architectures in such systems, and abstract these same statutes and architectures to form reusable information, which also reflects the essential requirements of the system.

Domain-specific software reuse activities can be better applied to domain engineering. Domain analysis is the basis of domain engineering, its main purpose is to obtain the domain model, and the role of the domain model is to describe the common requirements in the domain. This behavior needs to consider factors such as system requirements and changes, determine the appropriate scope, and clarify the common and variable characteristics in the field. FODM [28] is a feature-oriented domain modeling method proposed by Zhang Wei et al. This method mainly discusses the Similar to the modeling process of the domain algorithm, the general structure of this method is shown in Figure 1.

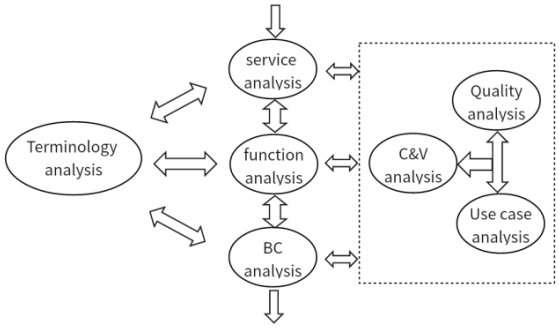

Figure 1. Domain modeling process.

It takes the service, function, and behavior characteristics of the domain as the key analysis objects, and at the same time collects the interaction situation, non-functional requirements, and related use cases in the

domain for analysis. The follow-up work of this paper will use the above method to analyze the BSAM field.

2.2. PAR Method

PAR is a unified framework for the formal development of algorithms. It is automatically converted from the generic specification and algorithm modeling language Radl, the transformation rules of the formal specification, the systematic algorithm programming methodology, the generic abstract programming language Apla, and a series of algorithms and program models. Tool composition. The PAR method makes full use of functional abstraction and data abstraction, and utilizes the top-down progressive refinement program development method to design abstract algorithms based on abstract functions and data, and gradually refine them until they can be developed using executable language statements. Using the PAR method to develop algorithms can deepen our understanding of algorithms, let us have a further understanding of algorithm design strategies, design and implement highly reliable components, and avoid usage errors. Apla can directly use abstract data types and abstract procedures to write programs, which is one of its major features. The form is concise, highly abstract, and the expression is unambiguous. Apla includes concepts such as identifiers, keywords, standard procedures, functions, type systems, symbolic expressions, program structures, statements, procedures, and abstract data types (abatract data type, ADT). Use sometype to represent type variables, someproc to represent process variables, and somefunc to represent function variables, which not only supports type parameterization, but also supports process parameterization and function parameterization.

The PAR method also includes a platform converter that supports the automatic conversion of the generic abstract programming language Apla into executable high-level programming languages such as JAVA, C++, and DELPHI, and also becomes the PAR platform. In the follow-up, the generic abstract programming language Apla in the PAR method will be used to realize the components and their dependencies in the interaction model, and then the component assembly algorithm will be used to convert it into an executable program by using the PAR platform.

3. Design and Implementation of BSAM Domain Model

3.1. Related Concepts of Approximate String Matching

Approximate string matching is string matching that allows "errors" to occur [29]. Given two strings, a text string T and a pattern string S, the characters in which all exist in a given character table Σ, the number of "errors" between the two strings is required to be within the given "error" Within the range, the "error" here can be represented by edit distance, Hamming distance, fragment distance, etc. The most common representation is to use the edit distance.

3.1.1. Edit distance

Edit distance, also known as Levenshtein distance [30], was proposed by Vladimir Levenshtein in 1965 to measure the similarity between two strings. Given strings s and t, the meaning of edit distance is: the minimum number of editing operations to transform string s into t. There are three modes of operation: insert, delete and replace. Generally, ed (s,t) is used to represent the edit distance between strings s and t.

3.1.2. Approximate string matching

Given a string set m (the elements in the set are denoted as t) and a query string s, the characters in it belong to a character set Σ, determine an edit distance threshold τ, and the set m meets ed(s,t)$\leqslant$ The element t of τ is the desired result, namely $\{t|t, s\in\Sigma, ed(s, t)\leqslant\tau\}$.

3.2. Algorithm Analysis and Modeling in Bsam Domain

3.2.1. Domain analysis

In this section, the core ideas of the three classic algorithms will be analyzed, and the gene sequence is used as a demonstration here.

(1) Q-gram algorithm

Q-gram is the original threshold-based approximation algorithm proposed in 1991. Its main process is as follows:

①Input a gene sequence *s* and a gene sequence set *m*, judge the legitimacy of the sequence *s* and set *m*, if wrong, the output is that the input information is wrong;

②Divide the gene sequence *s* by *q* to obtain gram features, set the division value to *q*, and determine an edit distance threshold τ;

③Perform *q* division on the given sequence set *m* to obtain the gram feature, and set the division value and the division value of the gene sequence *s* to the same value;

④Establish an inverted index for the gram features obtained after dividing the sequence set *m*;

⑤Calculate the number and type of each gram of the sequence *s* in the inverted index, and then count the number of occurrences of each element *t* by counting the number of occurrences of the gram feature of each element *t* in the string set *m*;

⑥If the number of occurrences of *t* is greater than or equal to $\max\{|s|,|t|\}-q+1-q*\tau$, the sequence *t* that initially satisfies the condition is obtained. Perform the ⑥ process on all the elements in the set to get the set *n*. Where $|s|$ represents the length of the sequence, and the function of max is to find the maximum value;

⑦ Perform online approximate matching of the sequence in the set *n* and the sequence *s* in the way of dynamic programming, and obtain the final result.

(2) VGRAM

The VGRAM algorithm is a method based on variable-length feature filtering proposed by Chen Li et al. in 2008. The basic process is as follows:

①Check the sequence validity of the input sequence *s* and sequence set *m*, and return an error message if there is an error;

②Use the VGEN algorithm to convert the string *s* into a gram feature set VG(*s*) with variable positions, and determine the edit distance threshold τ;

③In the same way, the element *t* in the sequence set *m* is converted into a gram feature with variable length;

④Establish an inverted index for the gram feature set of the sequence set *m*, and use the index structure to pre-calculate VG (*si*) and NAG(*si*). NAG (*si*) is the NGA vector of *si*, that is, for each string in the set, It is necessary to calculate how many grams will be affected by *k* editing operations;

⑤The common gram of the two sequences needs to be greater than or equal to max(|VG(*s*)|-NAG(*s*,*k*),|VG(*t*)|-NAG(*t*,*k*)) to get the set *n*;

⑥ For the sequence set *n*, perform an approximate matching algorithm in the online dynamic programming mode on each element in the *n* set and the sequence *s* to obtain the final result.

(3) Ed-Join

Ed-join is an algorithm based on mismatched filtering methods, and its basic process is as follows:

①Judging the legitimacy of the input gene sequence *s* and gene sequence set *m*, if it is wrong, the output and input are wrong;

②Divide the sequence set *m* into equal lengths of length *q* to obtain the feature set of gram, and determine the edit distance threshold τ;

③Take $q^*\tau+1$ prefix grams of two strings *s* and *t*. If these prefixes are all different and $\tau<\lceil(|s|-q+1)/q\rceil$, these sequences will not meet the requirements, and the remaining The sequences form a set *n*;

④ For each element t in the set *n*, select all the mismatched grams with the sequence *s* to form an array, and calculate the *L1distance* between the two mismatched grams, and then find the minimum modification times of all grams of the rightmost character *min-err*, if *L1distance*/2+1 is greater than τ, the element does not meet the requirements. After traversing the set *n*, the remaining element group set *n2*;

⑤Match each element *t* in the sequence set *n2* with the sequence *s* in an online approximate string matching in the way of dynamic programming, and obtain the final result.

Combining the above three algorithms and various other BSAM algorithms, we can analyze that, for the BSAM algorithm, it is first necessary to determine an edit distance threshold τ and a similarity comparison function Sim. Then perform feature extraction on the set *m* and the character string *s* to be matched. The value of the metric function is obtained through the Sim function. When the difference meets the requirements, a string result set including a certain error is initially obtained. Finally, online approximate string matching needs to be performed on this result set to obtain a string that fully meets the requirements. We can express the idea of these algorithms using a unified flowchart, as shown in Figure 2.

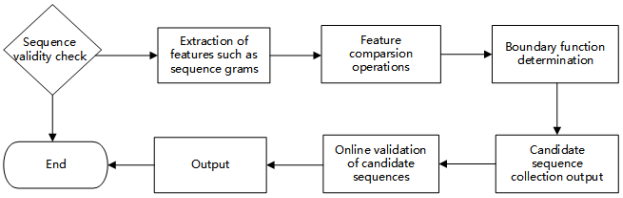

Figure 2. Flow chart of BSAM algorithm.

3.2.2. Domain modeling

Based on the above analysis, we obtain a unified algorithm flow in the BSAM field, and next, this paper will use the feature-oriented modeling method FODM proposed by Academician Mei Hong's team to model the features of the Service S, Function F and Behavior characteristics B in the BSAM field.

The core service in the BSAM domain is the sequence approximation matching service (sequence_approximate_match), which serves as the service layer of the feature model.

For the functional layer, according to the algorithm flow structure of BSAM, the sequence_approximate_match service contains seven functions, namely: sequence legitimacy check (seq_check), feature extraction operation (fea_extra_mani), index establishment operation (index_cre_mani), feature comparison (fea_compar), boundary function determination (boundary_fun), Online verification algorithm (online_verify), output (output). Because some algorithms do not use an index structure, indexing operations (index_cre_mani) are made optional and the rest are required.

For the behavioral characteristics layer, the behavioral characteristics of different functions are mainly analyzed, and other functions will be reusable as common features. In the functional layer, there are mainly different behavioral characteristics of feature extraction operation (fea_extra_mani) and feature comparison (fea_compar), and for fea_extra_mani functions, there are overlapping equal-length gram division algorithm (q-gram), variable-length gram division algorithm (VGEN), unequal length non-overlapping gram division algorithm (tail_restricted) and extraction sequence content (content) four main behavioral characteristics, the first three behavioral characteristics are multiple choice of one way, and the last behavioral characteristic can be combined with one of the first three behavioral characteristics; for the fea_compar function, there are five behavioral characteristics of count, position, prefix, length, and non-matching (mis_match), of which count, prefix, and non-match mis_match are comparison operations for features, position and length is a comparison operation against the original sequence. Based on the above analysis, the characteristic model of the BSAM domain is established as shown in Figure 3.

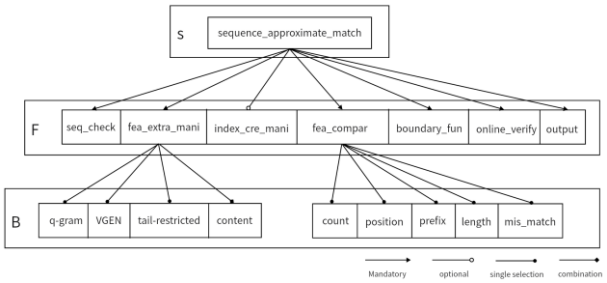

Figure 3. BSAM feature model.

In the above feature model, seq_check, index_cre_mani, boundary_fun, online_verify, output are common features in the field, fea_extra_mani, fea_compar are the different features in the field, where the main features that will be reused are the common features in the feature model, which will be reflected in the Apache program design.

In order to establish the corresponding components and the Apla program, it is also necessary to establish the interaction model between features, and the interaction between features is mainly reflected in the dependence and mutual constraints between features. Therefore, based on the previously established algorithm unified flowchart and feature model, this paper designs a feature interaction model between BSAM.

In order to fully represent the entire interaction model, two components are added here to directly compare operations (compar_mani) and intermediate result sets (mid_result). Analyze the constraints and dependencies between features, and establish the interaction model of algorithm components. The algorithm mainly includes two change process characteristics: fea_extra_mani and fea_compar. In addition, this paper takes the seq_check, boundary_fun, online_verify, and output in the feature model as the main components, and other features as related data structures and auxiliary components, and then the priority and dependencies between the components establish the interaction model of the components, as shown in Figure 4.

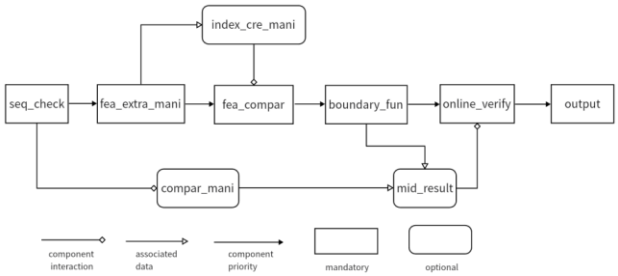

Figure 4. BSAM component interaction model.

In Figure 4, the features connected by the black triangular arrows are required for the BSAM domain, and the arrows show the execution priority of the features from high to low; the white triangular arrow indicates the transmission of data, for example: after the feature extraction operation (fea_extra_mani), the corresponding features of the sequence will be obtained, and these feature data will be transmitted to the index creation operation (index_cre_mani) for indexing; white prismatic arrows indicate interactions between components, for example, feature comparison (fea_compar) requires the use of features in the index in an index creation operation (index_cre_mani). Here is a summary of the component model, the common part of the domain can be summarized into 3 components, 3 components constitute 1 process:

(1) The first component is to obtain the sequence characteristics, that is, the results obtained by fea_extra_mani components, which includes seq_check components, fea_extra_mani components and index_cre_mani components;

(2) The second component is the data generated to obtain the intermediate candidate string, that is, the component boundary_fun and the component compar_mani, the process includes fea_compar component, boundary_fun component, compar_mani member;

(3) The third component is to obtain the final result, that is, to obtain the final result through verification, and the process includes online_verify components and out_put components.

The three components are summarized into a general process sequence_approximate_match. The above artifacts will serve as reusable artifacts in the BSAM domain. For the differences in the field, it is necessary to extract the feature extraction algorithm and boundary function that need to be used in the fea_extra_mani component and the boundary_fun component as a component to create it. These components will be represented as process parameters in the generic operation in the following Apla program.

4. Apla Program Implementation and Experimentation

Based on the analysis of feature model and component model in the previous section, this paper encapsulates five abstract data type (ADT) components and one sequence_approximate_match process. It was designed and implemented using Apla, a generic abstract programming language that uses the formal PAR method. AQ and AR are keywords defined in the Radl specification language to describe pre-assertion and post-assertion, respectively. Due to space limitations, only the four main common components are described in the Radl protocol.

4.1. Types and Algorithm Component Design

4.1.1. Sequence feature acquisition type component

Define ADT sequence_feature (someproc feature)

Function get_sequence_stream (filepath: array of character);

Function check (sequence:array of character):array of character;

Function get_feature (inSequence:array of character; feature_m:someproc):map;

Procedure create_index (get_feature: somefunc);

Enddef*;*

The name of the ADT is sequence_feature, which is a common component, which contains a process parameter feature, which is used to pass characteristic data, that is, the differential operation is input as a parameter, and the similar common component below also serves this role. Its function is to acquire sequence features. The role of the get_sequence_stream is to read the sequence as a file

stream; the function of check is to check whether the read sequence is legal; the function of the get_feature is to obtain sequence features, and the feature acquisition algorithm is used as the operating parameter of the function; the role of the create_index is to create an index.

Its Radl protocol:

|[in s[0:x]: array of character, p[0:y][0:z]: array of character, feature: procedure; out fea[0:i]: array of set]|

AQ: $x \geqslant 0 \wedge y \geqslant 0 \wedge z \geqslant 0$

AR: $i \geqslant 0$

The process of entering the two strings to be matched, and entering which features are extracted, requires that the number of sequences and the length x, y, and z need to be greater than or equal to 0. The output result is an array of features, the number of which must be greater than 0.

4.1.2. Feature acquisition components

Define ADT feature

Procedure q-gram (inSequence, querySequence:array of character; q:integer);

Procedure vgen (inSequence, querySequence: array of character; minq, maxq: integer);

Procedure tail_restricted (inSequence, querySequence: array of character);

Procedure content (inSequence, querySequence: array of character);

Enddef;

The ADT is called feature, which contains a series of generic subroutines to obtain different characteristics of the sequence, and the process will be passed as process parameters in other common constructions.

4.1.3. Intermediate candidate result type component

Define ADT mid_result (somefunc boundary_fun)

Function compare_feature (r_index: procedure; boundary_fun: somefunc): integer;

Function compare_sequence (inSequence, querySequence: array of character): boolean;

Function compar (result: integer; boundary_fun: somefunc): set;

Enddef;

The name of this ADT is mid_result, which contains a procedure parameter boundary_fun that is used to pass different boundary functions. Its purpose is to obtain intermediate results that are initially filtered, but there may still be errors. The function compare_feature is to compare features, and internally index establishment is used as a process parameter. The role of the compare_sequence is to compare sequences for non-feature comparisons, returning true when the sequence meets the requirements, otherwise returning false; The function compar is used to make a judgment on the result after the feature comparison, and if it is within the boundary, the sequence is stored in the set.

Its Radl protocol:

|[in: fea_s[0:x]: array of set, fea_p[0:y]: array of set; out: value:integer]|

AQ: $x \geqslant 0 \wedge y \geqslant 0$

AR: value:=($\forall$ value: $0 \leqslant i \leqslant x \wedge 0 \leqslant j \leqslant y \wedge$ fea_s[i]==fea_p[j]: value++)

The input is the feature sets fea_s and fea_p of strings, the number of feature sets is required to be greater than or equal to 0, and the output is the value returned after feature comparison.

4.1.4. Boundary function type component

Define ADT boundary_fun

Function compute_count_boundary(q,editDistDef:integer):integer;

Function compute_miscontent_boundary(inSequence,querySequence:array of character):integer;

Function compute_length_boundary(inSequence,querySequence:array of character;editDistDef,q:integer):integer;

Function compute_prefix_boundary(inSequence,querySequence:array of character;editDistDef,q:integer):integer;

Function compute_location_boundary(inSequence,querySequence:array of character;editDistDef,q:integer):integer;

Enddef;

The name of the ADT is boundary_fun, which contains various generic subroutines to compare different types of features, and the functions in this ADT will be passed as function parameters in other common components.

4.1.5. Final result components

Define ADT final_result

Function verify (mid_result: set; inSequence: array of character; editDistDef: integer): set;

Function computeLevenstein (inSequence, querySequence: array of character): integer;

Procedure output (result: set);

Enddef;

The name of this ADT is final_result. Its role is to obtain a sequence that exactly meets the requirements. The role of verify is to perform online sequence approximation matching of the obtained candidate results in a dynamic programming manner. The role of output is to output the final result.

Its Radl protocol is:

|[in: mid_p[0:x]: array of set, r: integer; out: final_p[0:y]: array of set]|

AQ: $0 \leqslant x \wedge 0 \leqslant r$

AR: $y \leqslant x$

The input is an intermediate candidate result set mid_p and the editing distance threshold r, and the output is the final result set final_p.

4.1.6. The sequence approximation matching process

Procedure sequence_approximate_match (sequence_feature: ADT; mid_result: ADT; final_result: ADT);

Its Radl protocol is:

|[in: s[0:x]: array of character, p[0:y][0:z]:array of set, fea: someproc, boundary: somefunc, r: integer; out: f[0:i][0:j]: array of set]|

AQ: $0 \leqslant x \wedge 0 \leqslant y \wedge 0 \leqslant z \wedge 0 < q \wedge 0 < r$

AR: $i \leqslant y \wedge 0 \leqslant j \wedge ed(s[x],p[y][z]) \leqslant r$

The input is pattern string S, string set P, feature extraction process FEA, boundary function boundary, and edit distance threshold R. The output is an array collection, which requires that the editing distance obtained be less than the predetermined editing distance, and the number of elements that conform to the string set p must be greater than or equal to the number of elements of the final result set.

4.2. Misq Algorithm Assembly

In this paper, the feature acquisition method and count of q-gram and content are combined with non-matching feature comparison, and they are assembled into an algorithm, named Misq.

```
Program Misq;
var
inSequence,querySequence:array of character;
q,editDistDef:integer;
databasemap:map(integer,array of character);
databasemap = open(“D:/match/Misq/source_sequence.fasta”);
indexList = databaseMap.keySet();
begin
forach(i=0;i<=indexList.length();i++)
querySequence := indexList.get(i);
.....//Program snippet, omitted
end;
procedure Misq_q-gram:new feature. q-gram (inSequence,querySequence,q); .....①
procedure Misq_content:new feature. content (inSequence,querySequence); .....②
ADT Misq_sequence_feature: new sequence_feature (Misq_q-gram,Misq_content); .....③
procedure Misq_compute_count_boundary: new boundary_fun. compute_count_boundary (q,editDistDef); ④
procedure Misq_compute_miscontent_boundary: new boundary_fun. compute_miscontent_boundary (inSequence, querySe-quence); .....⑤
ADT Misq_mid_result: new mid_result (Misq_compute_count_boundary, Misq_compute_miscontent_boundary); .....⑥
ADT Misq_final_result: new final_result(); .....⑦
procedure Misq_sequence_approximate_match: new sequence_approximate_match (Misq_sequence_feature, Misq_mid_re-sult, Misq_final_result); .....⑧
begin
Misq_sequence_approximate_match (Misq_sequence_feature, Misq_mid_result, Misq_final_result);
end;
```

In the above Apla program, code blocks ① and ② are the feature acquisition methods of the Misq algorithm; ③ acquisition of sequence features; ④, ⑤ is a feature comparison method; ⑥ to obtain intermediate or selective results; ⑦ to obtain the final result; ⑧ main procedure.

4.3. Experiments

Computer configuration: processor is Intel(R) Core(TM) i7-10700 CPU @ 2.90GHz 2.90 GHz, RAM: 8.00GB, operating system: Windows 10.

Genbank is a DNA sequence database established by the National Center for Biotechnology Information (NCBI), this paper selects the sequence numbered EU660217 and the length of 16283bp from Genbank for experiments, and compares the algorithm Misq and q-gram algorithm generated by assembly in a unified data set. Set the editing distance threshold to 5 and the division value to 3.The length of the substring is 30.The number of pattern strings is 1, 10, 30, 50, 70, 90,and 110,and the sequence length in the query sequence collection is fixed to 30 characters. The experimental results are shown in Figures 5 and 6.

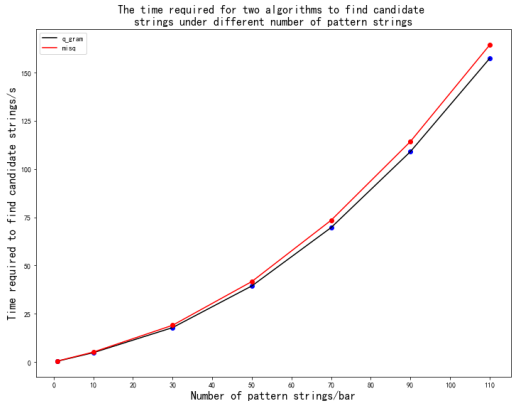

Figure 5. Comparison of candidate string time between the two algorithms

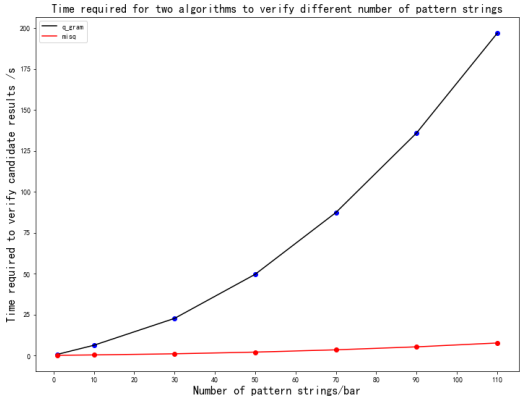

Figure 6. Comparison of the verification string time of the two algorithms.

As can be seen from Figure 5, the time spent by the two algorithms to find the candidate set is very close in feature comparison, and the time spent by the Misq algorithm is slightly more than that of the q-gram algorithm. Figure 6 shows the time taken by the q-gram algorithm and the Misq algorithm for final verification, and the verification algorithms used by the two algorithms are consistent, and it is clear that the time spent by Misq increases with the number of pattern strings, and the time spent for verification is much less than that of the q-gram algorithm.

It can be seen that the Misq algorithm generated by assembly has certain practicality.

5. Concluding Remarks

BSAM is an important algorithm in biological sequence analysis, often used to compare DNA or protein sequences of different species to study the relationship between sequences and the function of sequences. However, there are still some problems in this work in the biological field, the BSAM algorithm has the lack of filtering criteria, each filtering criterion is limited to each algorithm and cannot be expanded, the filter area selection is complex or simple, and the division of characteristics and filtering means require personalization. Therefore, it is necessary to study these algorithms from the level of the algorithm field.

This paper adopts the feature-oriented modeling method FODM, extracts the common features and heterogeneous features in the field, considers the main features such as feature extraction operation (fea_extra_mani), feature comparison (fea_compar), boundary function determination (boundary_fun), online verification algorithm (online_verify), etc., and analyzes and designs the interaction model between the feature model and its components in the BSAM field. And the generic abstract program language Apache using the high-confidence formal PAR method is implemented, and by custom-assembling the algorithm of these components, the implementation process of the approximate matching algorithm of this biological sequence is optimized, the reliability and abstraction of the algorithm is improved, and finally the series of programs of the PAR platform are converted into executable programs. Let people focus on the research and development of the heterogeneous characteristics of the algorithm, reduce the attention to the redundant part of the algorithm, realize personalized needs, deepen people's understanding of the algorithm, reduce the workload of algorithm development, and help people better choose the algorithm suitable for their own problems.

The easy verification of the Apla language ensures the reliability of assembly algorithms, and feature-oriented domain modeling can well extract the commonality and heterogeneity between various similar algorithm domains, make full use of the reusable components between these algorithm domains, improve development efficiency, and help people understand algorithms. The Apla language has a good high abstraction, representing the algorithm components, laying the foundation for the realization of the personalized requirements of the algorithm. The various methods and ideas in the field of BSAM in this paper are not only very applicable to biological sequence approximation matching algorithms, but also theoretically can provide ideas for algorithm research in other fields in bioinformatics, and the next step will continue to improve the algorithm component library, for a variety of newly researched BSAM algorithms, study their structural principles, and add new components for the BSAM field algorithm component library.

References

[1] Rusk N. Cheap third-generation sequencing. *Nature Methods*, **2009**, 6(4): 244-244.

[2] Lipman D J, Pearson W R. Rapid and sensitive protein similarity searches. *Science*, **1985**, 227(4693): 1435-1441.

[3] Altschul S F, Gish W, Miller W, et al. Basic local alignment search tool. *Journal of molecular biology*, **1990**, 215(3): 403-410.

[4] Jokinen P, Ukkonen E. Two algorithms for approximate string matching in static texts//*International Symposium on Mathematical Foundations of Computer Science*. Springer, Berlin, Heidelberg, **1991**: 240-248.

[5] Ma B, Tromp J, Li M. PatternHunter: faster and more sensitive homology search. *Bioinformatics*, **2002**, 18(3): 440-445.

[6] Navarro G. A guided tour to approximate string matching. *ACM computing surveys (CSUR)*, **2001**, 33(1): 31-88.

[7] Smith T F, Waterman M S. Identification of common molecular subsequences. *Journal of molecular biology*, **1981**, 147(1): 195-197.

[8] Yang X, Wang B, Li C. Cost-based variable-length-gram selection for string collections to support approximate queries efficiently//*Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. **2008**: 353-364.

[9] Bayardo R J, Ma Y, Srikant R. Scaling up all pairs similarity search//*Proceedings of the 16th international conference on World Wide Web*. **2007**: 131-140.

[10] Li C, Wang B, Yang X. VGRAM: Improving Performance of Approximate Queries on String Collections Using Variable-Length Grams//*VLDB*. **2007**, 7: 303-314.

[11] Yang X, Wang B, Li C. Cost-based variable-length-gram selection for string collections to support approximate queries efficiently//*Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. **2008**: 353-364.

[12] Wang W, Qin J, Xiao C, et al. VChunkJoin: An Efficient Algorithm for Edit Similarity Joins. *IEEE Transactions on Knowledge & Data Engineering*, **2013**, 25(8):1916-1929.

[13] Burrows M, Wheeler D. A block-sorting lossless data compression algorithm. *Digital SRC Research Report*, **1994**.

[14] Heng Li, Richard Durbin, Fast and accurate short read alignment with Burrows–Wheeler transform, *Bioinformatics*, Volume 25, Issue 14, July **2009**, Pages 1754–1760

[15] Esko Ukkonen. Approximate String-Matching over Suffix trees. Proc. CPM 93, *Lecture Notes in Computer Science 684*, Springer **1993**, 228—242.

[16] R.A. Baeza-Yates, GH. Gonnet. Fast string matching with mismatches. *Information and Computation*, 1994, 108(2):187—199.

[17] Ma nber U, Myerst A G. Suffix arrays: A new method for on-line string searches. *Siam Journal on Computin*g, **1993**, 22 (5):935-948.

[18] Sun Decai,Sun Xingming,Liu Yuling.A Filter Algorithm for Approximate String Matching Based on Match-Region Features. *Journal of Computer Research and Development*, **2010**, 47(04):663-670.

[19] Wang jiaying, Wang Bin, Yang Xiaochun. A Space Efficient Approach of Multicore Parallel Approximate Substring Matching. *Journal of Computer Research and Development*, **2015**, 52(S1):37-47.

[20] Navarro G, Raffinot M. Flexible pattern matching in strings: practical on-line search algorithms for texts and biological sequences. *Cambridge university press*, **2002**.

[21] Li Keqin, Chen Zhaoliang, Mei Hong, Yang Fuqing. An Outline of Domain Engineering. *Computer scinence*, **1999**(05):21-25.

[22] Xue J. A unified approach for developing efficient algorithmic programs. *Journal of computer Science and Technology*, **1997**, 12(4): 314-329.

[23] Wang C J, Xue J Y. Formal derivation of a generic algorithmic program for solving a class of extremum problems//Proc of the 2009 10th ACIS International Conference on Software Engineering, *Artificial Intelligence*, Networking and Parallel/Didtributed Computing, **2009**: 100-105.

[24] Xue J. Genericity in PAR platform// Structured Object-Oriented Formal Language and Method. *Cham:Springer*, **2015**:3-14.

[25] Xue J. Two new strategies for developing loop invariants and their applications. *Journal of Computer Science and Technology*, **1993**, 8(2):147-154.

[26] Xue J. Formal derivation of graph algorithmic program susing partition-and-recur. *Journal of Computer Science and Technology*, **1998**, 13(6):553-561.

[27] Xue J Y, Zheng YJ, HuQ M, et al.P AR: A practicable formal method and its supporting platform// *Formal Meth-ods and Software Engineering. Cham: Springer International Publishing*, **2018**: 70-86.

[28] Zhang Wei, Mei Hong. A Feature-Oriented Domain Model and Its Modeling Process,**2003**(08):1345-1356.DOI:10.13328/j.cnki.jos.2003.08.001.(in Chinese)

[29] Navarro G A guided tour to approximate string matching. *ACM Computing Surveys*, **2001**, 33(1): 31-88.

[30] Navarro, Gonzalo. A guided tour to approximate string matching (PDF). *ACM Computing Surveys*. 1 March **2001**, 33 (1): 31–88 [19 March 2015]. doi:10.1145/375360.375365.