BIOINFORMATICS

Vol. 19 no. 11 2003, pages 1391–1396
DOI: 10.1093/bioinformatics/btg168

Alignment of BLAST high-scoring segment pairs based on the longest increasing subsequence algorithm

Hongyu Zhang*

Celera Genomics, 45 West Gude Drive, Rockville, MD 20850, USA

Received on September 16, 2002; revised on January 2, 2003; accepted on February 14, 2003

ABSTRACT

Motivation: The popular BLAST algorithm is based on a local similarity search strategy, so its high-scoring segment pairs (HSPs) do not have global alignment information. When scientists use BLAST to search for a target protein or DNA sequence in a huge database like the human genome map, the existence of repeated fragments, homologues or pseudogenes in the genome often makes the BLAST result filled with redundant HSPs. Therefore, we need a computational strategy to alleviate this problem.

Results: In the gene discovery group of Celera Genomics, I developed a two-step method, i.e. a BLAST step plus an LIS step, to align thousands of cDNA and protein sequences into the human genome map. The LIS step is based on a mature computational algorithm, Longest Increasing Subsequence (LIS) algorithm. The idea is to use the LIS algorithm to find the longest series of consecutive HSPs in the BLAST output. Such a BLAST+LIS strategy can be used as an independent alignment tool or as a complementary tool for other alignment programs like Sim4 and GenWise. It can also work as a general purpose BLAST result processor in all sorts of BLAST searches. Two examples from Celera were shown in this paper.

Contact: me@hongyu.org

INTRODUCTION

Bioinformatics tools used in human genome study often have one or both of the following functions: database search or sequence alignment. Database search tools like BLAST (Altschul *et al.*, 1990, 1997) and FASTA (Pearson and Lipman, 1988) are used to search from a sequence database for the possible close relatives of a query sequence, while alignment tools like CLUSTALW (Higgins *et al.*, 1994), Sim4 (Florea *et al.*, 1998), and geneWise (Birney and Durbin, 1997) are used to find the best alignment between two sequences or multiple sequences.

There are already quite a number of programs designed to align a protein or transcript sequence to a genomic sequence. Sim4 (Florea *et al.*, 1998), Est2gen (Birney and Durbin, 1997), est_genome (Mott, 1997) and Spidey (Wheelen *et al.*, 2001) were programmed to align an mRNA sequence to a genomic sequence, and programs like geneWise and estWise (Birney and Durbin 2000) can be used to align a protein sequence with a genomic sequence. These programs are prohibited by their speed to be used as database search tools in the human genome study. More efficient programs like BLAST have to be used to search against the whole human genome component database. BLAST can run a search of a typical length cDNA sequence against the whole human genome map within tens of seconds using modern computers. Some other programs like SSAHA (Ning *et al.*, 2001) and BLAT (Kent, 2002) were published recently, and they can search the human genome database in a faster speed than BLAST because they use a different indexing strategy from BLAST and a longer word size than the standard BLAST program, which is a strategy also adopted by Mega-BLAST (Zhang *et al.*, 2000). Compared to them, BLAST still keeps its advantage in sensitivity and flexibility.

The BLAST program also has an obvious deficiency. Since it is in principle a local similarity search program, its output often contains many redundant HSPs. Usually it is because of the existence of homologues, pseudogenes or some repeated fragments in the genome. In a simple situation illustrated in Figure 1, we have to visually scan the BLAST output file to find the correct consecutive HSP list. The redundancy can cause two problems for scientists. First, in lots of situations, it is not always easy to find the correct longest consecutive list of HSPs if there are many HSPs in the results, and the task can become overwhelming for human eyes if there are many protein or cDNA sequences to be processed. Second, if there are multiple genomic component hits with similar significant

*Present address: Ceres Inc., 3007 Malibu Canyon Road, Malibu, CA 90265, USA.

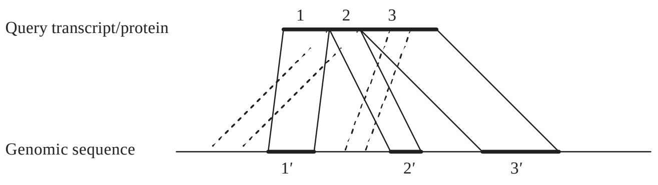

Fig. 1. A typical BLASTN result. Segment 1, 2, and 3 are consecutive segments in the query sequence, and their matches in the genomic sequence (1′, 2′ and 3′) are also consecutive. Although segments 2 and 3 also have other matches in the genomic sequence, segment (1, 2, 3) matching with (1′, 2′, 3′) consist the longest consecutive HSP list.

Table 1. The header of the BLAST output

```
BLASTN 2.1.2 [Nov-13-2000]


Reference: Altschul, Stephen F., Thomas L. Madden, Alejandro A. Schaffer,
Jinghui Zhang, Zheng Zhang, Webb Miller, and David J. Lipman (1997),
"Gapped BLAST and PSI-BLAST: a new generation of protein database search
programs",  Nucleic Acids Res. 25:3389-3402.

Query= CRA|11000571340367 /trans_lig_id=CRA|341000000002106
/dataset=LTI_cDNA rndm_seq /def=asm|341000000002106.Contig2 /org=Homo
sapiens /date=12/27/2000 /clone_id=CRA|19600411863039
/tissue=leukocyte
         (3220 letters)

Database: /work/gdisc2/pipeline_runs/latest_release/CHGD_assembly_late
st.fasta
           54,061 sequences; 2,908,729,056 total letters

Searching..................................................done

                                                                   Score     E
Sequences producing significant alignments:                        (bits)  Value

CRA|GA_x2HTBL3EHN0:1..2266668 /chromosome=9 /organism=Homo sapi...  2406  0.0
CRA|GA_x2KMHMRK2MR:1..11360698 /chromosome=14 /organism=Homo sa...   104  4e-19
CRA|GA_x54KRCCT8MR:1..3514905 /chromosome=13 /organism=Homo sap...   102  1e-18
CRA|GA_x54KRCC3F0N:1..5394345 /chromosome=15 /organism=Homo sap...    98  2e-17
CRA|GA_x2KMHMQRB1V:1..9064012 /chromosome=14 /organism=Homo sap...    98  2e-17
CRA|GA_x2HTBL41RAJ:1..12379226 /chromosome=18 /organism=Homo sa...    98  2e-17
CRA|GA_x54KRCCSU5M:1..6553263 /chromosome=17 /organism=Homo sap...    94  4e-16
CRA|GA_x2HTBKPMQBA:1..7485961 /chromosome=10 /organism=Homo sap...    94  4e-16
CRA|GA_x2HTBKLHUKM:1..5175465 /chromosome=1 /organism=Homo sapi...    90  6e-15
CRA|GA_x2HTBL5BKUY:1..7334859 /chromosome=12 /organism=Homo sap...    88  2e-14
```

scores, it is not straightforward to identify which genomic component contains the correct gene. Therefore, we need a computer program to help us.

In this work, I designed a two-step method that combined the strength of multiple programs to implement a tool that can be used in fast locating of a transcript or protein sequence in the human genome map. The central idea is to use the Longest Increasing Subsequence (LIS) algorithm to find the longest list of consecutive, non-overlapping HSPs in a BLAST output.

ALGORITHM

Our genome search method consists of two major steps. The first step in our method is to perform a genome scale BLAST search. In the second step, we use the LIS algorithm to find the longest list of consecutive, non-overlapping HSPs for each BLAST hit, and then do some redundancy filtering.

In the first step, we need to choose a correct BLAST program and appropriate program parameters. To search with an mRNA sequence against the human genome, we can use BLASTN or TBLASTX, and to search with a protein sequence, we use TBLASTN. In order to remove the fuzzy alignments that usually appear in the two edges of HSPs, we use big gap penalties in BLAST searches.

The LIS algorithm used in the second step is originally an algorithm to find the longest monotonically increasing subsequence in a sequence of n numbers (Gusfield, 1997; Skiena, 1997). Consider a sequence $S = (9, 5, 2, 8, 7, 3, 1, 6, 4)$, the longest increasing subsequence of S has length 3 and is either (2, 3, 4) or (2, 3, 6). There are two major implementations of LIS algorithm. The simpler version is a dynamic programming technique. Its time complexity is $O(n^2)$, where n is the number of HSPs. A more complicated but faster version has a time complexity of $O(r\log(n))$ (Gusfield, 1997).

LIS algorithm has been used in studying some similar problems in previous references. When aligning the sequences of two genomes, Delcher *et al.* (1999) used this algorithm to successfully extract the longest set of MUMs whose sequences occur in ascending order in both Genome A and Genome B, where MUM refers to the Maximal Unique Matches between two genomes. Although not using the LIS algorithm, programs like Sim4 (Florea *et al.*, 1998) or SALSA (Rognes and Seeberg, 1998) also contain a similar strategy of combining a fragment scanning stage plus a dynamic programming alignment stage to find candidate alignment. Comparatively, in our implementation, we directly use BLAST program to find the significant fragment matches, and then process them using the LIS algorithm.

As explained previously in Figure 1, the purpose of using the LIS algorithm in our work is to parse the BLAST output and find the longest list of non-overlapping HSPs with their positions in consecutive order in both the query and the subject sequence. Here are the details of how we implemented the LIS algorithm in a dynamic programming version. First, for each genomic hit we can pick up all n HSPs that are in the forward match direction and sort them in the increasing order based on their query positions (we will process all the reverse complimented HSPs separately later). The sorted HSPs are put in an array $\{hsp_1, hsp_2, \ldots, hsp_n\}$. We then define l_i as the LIS of the first i HSPs ($i = 1$ to n), where l_1 equal to $\{hsp_1\}$, and the rest LISs are derived based on a recursive relationship shown in Equation (1). In the recursive deduction, l_i has to end up with hsp_i. Such a constraint makes it possible to deduce l_i from all previous l_1 to l_{i-1}, which is well explained in the book of Skiena (1997).

$$\begin{cases} l_1 = \{hsp_1\} \\ l_i = \{\max\limits_{1\leqslant j<i} l_j, hsp_i\}, \quad if\ 0 \leqslant s_i - e_j < Cutoff. \end{cases} \tag{1}$$

In Equation (1), $\max_{1 \leqslant j<i} l_j$ represents the l_j that has the longest total HSP length among $\{l_1, l_2, ..., l_{i-1}\}$, where the total HSP length is defined to be the sum of the lengths of all component HSPs. s_i is the start position of hsp_i in the subject sequence, e_j is the end position of l_j in the subject sequence, and *Cutoff* is the maximal intron size selected approximately as 500 kb. The constraint of $0 \leqslant s_i - e_j < Cutoff$ is to enforce that the distance between any neighboring HSP pairs be greater or equal to zero, i.e. they are connected but not overlapped (in reality we relaxed the limit a bit to allow a maximum 10-base overlap that often appears in BLAST results) and less than a cutoff that is the maximal intron size of 500 kb.

After recursively finding all l_i, we can pick up the longest one among them, which is the final LIS that we were looking for, i.e.

$$LIS = \max_{1 \leqslant i \leqslant n} l_i. \tag{2}$$

In a similar way, we can find the longest list of HSPs in the reverse complimented direction. The final choice will be the longer one between the two. In case that there are multiple l_i having an equal total HSP length, the program will report all of them.

Another important problem faced by this method is that in lots of search results we have more than one genomic hit in the BLAST result, and every component hit covers either a partial or possibly the entire region of the query sequence. So we have to decide how to deal with those situations based on some rules that both make sense in biology and also are easy to implement in computer algorithm. We called those rules as *context-logic* because they are mainly based on the relationship between the multiple genomic component hits in the context of the coverage of the query sequence. What we used is a greedy approach, i.e. first we look for the LISs for each genomic component hit, so we rank all the genomic hits based on the length of the LISs. If the LIS of first hit cover the whole range of the query sequence, it means possibly a complete gene was found. For the rest hits, if their LISs only cover partial regions of the query sequence, most probably they only code for one of the homologues or a repeated fragment of the query sequence and we will discard them (we are going to discuss the potential dangers of this choice in the **Discussion** section). If the LIS of another hit also covers the whole region of the query sequence, we will keep it and consider it another possible coding HSP list.

Another scenario of multiple genomic hits is that the first hit only covers a partial region of the query sequence, which means that the gene is not complete in this genomic sequence and we need to find other genomic components that compliment the first component in the coverage of the query sequence. Our current method is to see whether the second hit covers a region of the query sequence that is not covered by the first hit: if so, it will be kept; otherwise, it will be discarded. The same procedure was applied for the third and the remaining hits, if they exist.

Table 2. Tabular HSP output*

```
Hit #1: CRA|GA_x2HTBL3EHN0:1..2266668
     cDNA_start cDNA_end   Component_start Component_end Identity
     1810       3012       814019          815221        1.00
     198        443        962492          962737        1.00
     1606       1810       820436          820640        1.00
     1          199        1015831         1016029       1.00
     1431       1608       824829          825006        1.00
     798        964        863297          863463        1.00
     964        1120       848724          848880        1.00
     3020       3173       813858          814011        1.00
     621        733        885420          885532        1.00
     1167       1273       838462          838568        1.00
     527        623        954014          954110        1.00
     1348       1433       827996          828081        1.00
     442        527        960798          960883        1.00
     1270       1349       830684          830763        1.00
     734        798        884462          884526        1.00
     1120       1175       839903          839958        1.00
     2741       2811       309713          309783        0.96
     2736       2783       2231644         2231691       0.98
     2740       2787       533921          533968        0.98
     2741       2787       2045402         2045448       0.98
     2736       2781       1660414         1660459       0.98
     2743       2787       169301          169345        0.98
     2743       2787       448997          449041        0.98
```

* Only the HSPs of the first genomic hit are displayed here because of space reason. The whole output can be viewed on the web at http://hongyu.org/paper/lis_example/blast1.tab.

RESULTS

I implemented this algorithm during my work in the gene discovery group of Celera Genomics at the end of 2000. It was used as a part of the Celera gene discovery pipeline. We combine this tool with other programs like Sim4 and GeneWise to annotate new cDNA clones to discover possible new gene targets. The program had processed thousands of cDNA and protein sequences before I left the gene discovery group to join another group of Celera in April of 2001.

I want to demonstrate the results of the program using two Celera cDNA sequence examples.

Table 1 displays the original BLAST output of a Celera cDNA sequence search against the Celera human genome assembly, in which there are 10 significant genomic component hits. Table 2 lists all the HSPs in the tabular format, which obviously contains lots of redundant HSPs in each hit. After applying the LIS program and *context-logic*, the result is shown in Table 3, from which we can see that the program not only found the longest consecutive HSP list by using the LIS algorithm, but also filtered out all the redundant genomic hits. The selected genomic hit covers nearly 100% of the query sequence, and the list of HSPs in the output gave a clear suggestion of the exon–intron structure.

Table 3. LIS output

```
Hit #1
-------
    Hit ID :                        CRA|GA_x2HTBL3EHN0:1..2266668
    Alignment direction :           reverse complement
    Aligned fraction of query :     99 %
    LIS #1 :
        cDNA_start cDNA_end   Component_start Component_end Identity
        1          199        1016029         1015831       1.00
        198        443        962737          962492        1.00
        442        527        960883          960798        1.00
        527        623        954110          954014        1.00
        621        733        885532          885420        1.00
        734        798        884526          884462        1.00
        798        964        863463          863297        1.00
        964        1120       848880          848724        1.00
        1120       1175       839958          839903        1.00
        1167       1273       838568          838462        1.00
        1270       1349       830763          830684        1.00
        1348       1433       828081          827996        1.00
        1431       1608       825006          824829        1.00
        1606       1810       820640          820436        1.00
        1810       3012       815221          814019        1.00
        3020       3173       814011          813858        1.00
```

A more interesting example is shown in Tables 4–6. The BLAST output has seven significant genomic hits shown in Tables 4 and 5. Our program filtered out all the genomic hits except for two genomic hits shown in Table 6. The two genomic components covered different parts of the query sequence. The first component covered almost the whole query sequence except for a part of its central region, while the second one covered exactly the central region of the query sequence. Very interestingly, the second hit is a short DNA fragment with unknown chromosome number, which means that the genome assembly team in Celera does not have enough proof to decide which chromosome this fragment should be put into. When we check the sequences of the two components, we found out that there is an N-gap region in the first component, and the second component, because of its small size, can right fit into the N-gap region of the first component, as illustrated in Figure 2. Moreover, these two components also share a same fragment of sequence of 513 bases, which is colored in gray in Figure 2 and is located upstream to the N-gap region of the first component and in the 5′ end of the second component. All these evidences strongly suggest that the second component is a missing part of the first component in the shot-gun assembly.

DISCUSSION

The method in this paper combines the advantages of two algorithms, BLAST and LIS. First of all, BLAST is very powerful for its speed, sensitivity and flexibility, so we can do almost all sorts of database search using BLAST. Then, the LIS algorithm is very quick and effective to pick up the most useful information from the BLAST output. Usually BLAST search takes the majority of the computing time, from tens of seconds to minutes to search for a typical length mRNA sequence against the whole human genome assembly, while the

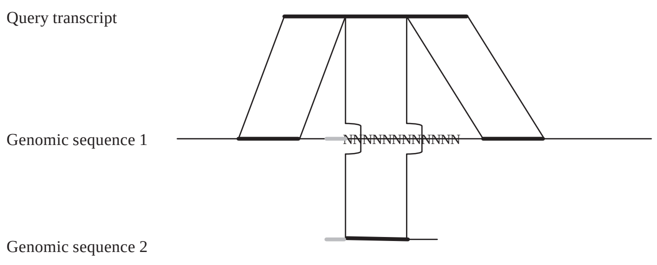

Fig. 2. One transcript is aligned with two genomic components. The second component happens to be able to fit in the middle N-gap region of the first big component. And the two components share a same 513 base-long sequence colored in gray.

Table 4. The header of the BLAST output*

```
BLASTN 2.1.2 [Nov-13-2000]


Reference: Altschul, Stephen F., Thomas L. Madden, Alejandro A. Schaffer,
Jinghui Zhang, Zheng Zhang, Webb Miller, and David J. Lipman (1997),
"Gapped BLAST and PSI-BLAST: a new generation of protein database search
programs",  Nucleic Acids Res. 25:3389-3402.

Query= CRA|11000567083186 /trans_lig_id=CRA|341000000001293
/dataset=LTI_cDNA rndm_seq /def=asm|341000000001293.Contig6 /org=Homo
sapiens /date=11/18/2000 /clone_id=CRA|19600411940083
/tissue=fetal_whole_brain
         (3442 letters)

Database: /work/gdisc2/pipeline_runs/latest_release/CHGD_assembly_late
st.fasta
           479,986 sequences; 3,281,761,131 total letters

Searching..................................................done

                                                                 Score     E
Sequences producing significant alignments:                      (bits)  Value

CRA|GA_x54KRCCNGEJ:1..2166028 /chromosome=13 /organism=Homo sap...   508  e-140
CRA|GA_x54KRE2F1CS:1..13238 /chromosome=U /organism=Homo sapien...   482  e-133
CRA|GA_x54KRCCNGAU:1..14052 /chromosome=U /organism=Homo sapien...   482  e-133
CRA|GA_x54KRDAVJLM:1..1095 /chromosome=U /organism=Homo sapiens...   126  9e-26
CRA|GA_x54KRDAVJLL:1..683 /chromosome=U /organism=Homo sapiens ...   126  9e-26
CRA|GA_x54KRDAVJLK:1..639 /chromosome=U /organism=Homo sapiens ...   126  9e-26
CRA|GA_x2HTBL2Y49L:1..2141229 /chromosome=6 /organism=Homo sapi...   126  9e-26
```

LIS algorithm takes multiple magnitudes less time. The current implementation of the LIS algorithm in this paper is a dynamic programming version written in the language PERL. Use of the second version of LIS implementation described in Gusfield (1997) will make it faster. And using C/C++ or other non-interpreted language rather than PERL will further improve the speed. However, since the majority of time in this method is spent on BLAST, it is not important for us at present to increase the speed of the LIS step.

Because of the roughness of the BLAST algorithm in the treatment of the exon–intron boundaries, the output could miss one or more exons in the genomic sequence even when the genomic sequence does contain the complete gene. In the real applications, we usually use Sim4 or GeneWise to make a further refinement of the alignment. Because we run the Sim4 or GeneWise on the specific genomic chunk that was located by the LIS algorithm, they can run very fast. We also add a default 2000 bases in both

Table 5. Tabular HSP output

```
Hit #1: CRA|GA_x54KRCCNGEJ:1..2166028
     cDNA_start cDNA_end   Component_start Component_end Identities
     1312       1565       1274276         1274529       1.00
     3205       3408       1223798         1224001       1.00
     127        321        1456409         1456603       1.00
     1561       1752       1260219         1260410       1.00
     1006       1169       1279634         1279797       1.00
     2780       2946       1231404         1231571       0.99
     3051       3208       1225416         1225573       1.00
     762        910        1283939         1284087       1.00
     1842       1977       1257709         1257844       1.00
     2292       2425       1236533         1236666       1.00
     1          127        1527239         1527366       0.99
     2663       2779       1232006         1232122       1.00
     647        761        1284910         1285024       1.00
     2945       3055       1226029         1226139       1.00
     322        428        1325988         1326094       1.00
     547        646        1291836         1291935       1.00
     908        1005       1281276         1281373       1.00
     1751       1842       1258921         1259012       1.00
     2493       2583       1233045         1233135       1.00
     2583       2666       1232198         1232281       1.00
     470        550        1301561         1301641       1.00
     1237       1314       1276035         1276112       1.00
     1165       1239       1278873         1278947       1.00
     2424       2493       1233369         1233438       1.00
     426        470        1304332         1304376       1.00

Hit #2: CRA|GA_x54KRE2F1CS:1..13238
     cDNA_start cDNA_end   Component_start Component_end Identities
     2053       2293       11455           11695         1.00
     1976       2055       4986            5065          1.00

Hit #3: CRA|GA_x54KRCCNGAU:1..14052
     cDNA_start cDNA_end   Component_start Component_end Identities
     2053       2293       12269           12509         1.00
     1976       2055       5800            5879          1.00

Hit #4: CRA|GA_x54KRDAVJLM:1..1095
     cDNA_start cDNA_end   Component_start Component_end Identities
     2424       2493       77              146           0.99

Hit #5: CRA|GA_x54KRDAVJLL:1..683
     cDNA_start cDNA_end   Component_start Component_end Identities
     2424       2493       175             244           0.99

Hit #6: CRA|GA_x54KRDAVJLK:1..639
     cDNA_start cDNA_end   Component_start Component_end Identities
     2424       2493       84              153           0.99

Hit #7: CRA|GA_x2HTBL2Y49L:1..2141229
     cDNA_start cDNA_end   Component_start Component_end Identities
     2424       2493       1729794         1729863       0.99
```

Table 6. LIS output

```
Hit #1
-------
    ID :                              CRA|GA_x54KRCCNGEJ:1..2166028
    Align direction :                 reverse
    Fraction of aligned query :       91 %
    LIS #1
        cDNA_start cDNA_end   Component_start Component_end Identity
        1          127        1527239         1527366       0.99
        127        321        1456409         1456603       1.00
        322        428        1325988         1326094       1.00
        426        470        1304332         1304376       1.00
        470        550        1301561         1301641       1.00
        547        646        1291836         1291935       1.00
        647        761        1284910         1285024       1.00
        762        910        1283939         1284087       1.00
        908        1005       1281276         1281373       1.00
        1006       1169       1279634         1279797       1.00
        1165       1239       1278873         1278947       1.00
        1237       1314       1276035         1276112       1.00
        1312       1565       1274276         1274529       1.00
        1561       1752       1260219         1260410       1.00
        1751       1842       1258921         1259012       1.00
        1842       1977       1257709         1257844       1.00
         <-- Gap -->
        2292       2425       1236533         1236666       1.00
        2424       2493       1233369         1233438       1.00
        2493       2583       1233045         1233135       1.00
        2583       2666       1232198         1232281       1.00
        2663       2779       1232006         1232122       1.00
        2780       2946       1231404         1231571       0.99
        2945       3055       1226029         1226139       1.00
        3051       3208       1225416         1225573       1.00
        3205       3408       1223798         1224001       1.00

Hit #2
-------
    ID :                              CRA|GA_x54KRE2F1CS:1..13238
    Align direction :                 forward
    Fraction of aligned query :       9 %
    LIS #1 :
        cDNA_start cDNA_end   Component_start Component_end Identity
        1976       2055       4986            5065          1.00
        2053       2293       11455           11695         1.00
```

the upstream and the downstream direction in the genomic sequence to do the Sim4 alignment to include the possible missing exons in the 5′ or 3′ end of the gene.

In most situations only the original genes are interesting to researchers, occasionally, however, people want to see some homologue or paralogue information. This can be achieved by an easy modification of the LIS algorithm in this paper: after finding the longest chain of HSPs that codes for the original gene, we can remove them and repeat the same procedure to find the remaining second or third longest chain of HSPs and so on, which are the candidate genes for those homologues and paralogues.

Although this paper only provides two examples because of the space limit, it does not mean that the program just works on anecdotal situations: Celera gene discovery group has used it to process thousands of new cDNA sequences. We understand that there are always exceptions in biology, and the method we used may not always pick up the real gene as the number one candidate. In case of uncertainties, such as when there are multiple genomic regions containing the same full gene, the program will always try to report all of them. Then human annotators can have their chance to carefully exam those hits using their expertise or other computational tools. The sole purpose of the algorithm is to help scientists reduce annotation effort and accelerate the discovery pace without sacrificing sensitivity.

One of the areas in my current method that needs to be improved in future is the way to implement *context-logic*. For example, in example 2, genomic hit 2 and genomic hit 3 cover exactly the same region of the query sequence, but genomic 3 was filtered out based on the current *context-logic* rules. Such a strategy will miss a possibly useful hit, genomic hit 3. To solve such a problem, we need to look for smarter methods, like a hash-based or suffix tree-based multiple genomic sequence alignment program. This work is still in development. In the Celera Genomics program described in this work, we just use simple factors such as the genomic component lengths and their chromosome numbers to improve the sensitivity of the *context-logic* filter.

It is important to mention that right after this job was done, which is between the end of 2000 and the beginning of 2001, some other programs like SSAHA (Ning *et al.*, 2001) and BLAT (Kent, 2002) were published. Their speeds, especially that of BLAT, are faster than BLAST. BLAT can also print an aligned tabular formatted HSP list that is similar to the LIS output in Table 6. Although inferior in the speed edge, the BLAST program still has its advantage and uniqueness in sensitivity and flexibility. For program SSAHA, the same combinational strategy can also be applied, e.g. SSAHA+LIS.

The program described in this paper can be used as a BLAST result processor also in other BLAST searches, so it is a more general purpose bioinformatics protocol for the scientific community. Even all the examples in this paper are from the Celera Genomics resource, the author did test the same program successfully on the data from public resources.

ACKNOWLEDGEMENTS

I am thankful to all my co-workers in the gene discovery group of Celera Genomics, especially to Ms Rhonda Brandon, Dr Maureen Higgins and Dr Ellen Beasley for their helpful advice and discussion on the design of the *context-logic* rules.

REFERENCES

Altschul,S.F., Gish,W., Miller,W., Myers,E.W. and Lipman,D.J. (1990) Basic local alignment search tool. *J. Mol. Biol.*, **215**, 403–410.

Altschul,S.F., Madden,T.L., Schaffer,A.A., Zhang,J., Zhang,Z., Miller,W. and Lipman,D.J. (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.*, **25**, 3389–3402.

Birney,E. and Durbin,R. (1997) Dynamite: a flexible code generating language for dynamic programming methods used in sequence comparison. *Proc. Fifth. Int. Conf. Intelligent Systems Mol. Biol.*, **5**, 56–64.

Delcher,A.L., Kasif,S., Fleischmann,R.D., Peterson,J., White,O. and Salzberg,S.L. (1999) Alignment of whole genomes. *Nucleic Acids Res.*, **27**, 2369–2376.

Florea,L., Hartzell,G., Zhang,Z., Rubin,G.M. and Miller,W. (1998) Computer program for aligning a cDNA sequence with a genomic DNA sequence. *Genome Res.*, **8**, 967–974.

Gusfield,D. (1997) *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology*. Cambridge University Press, New York.

Higgins,D., Thompson,J. and Gibson,T. (1994) CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Res.*, **22**, 4673–4680.

Kent,W.J. (2002) BLAT—the BLAST-like alignment tool. *Genome Res.*, **12**, 656–664.

Lander,E.S., Linton,L.M., Birren,B., Nusbaum,C., Zody,M.C., Baldwin,J., Devon,K., Dewar,K., Doyle,M., FitzHugh,W. *et al.* (2001) Initial sequencing and analysis of the human genome. *Nature*, **409**, 860–921.

Mott,R. (1997) EST_GENOME: a program to align spliced DNA sequences to unspliced genomic DNA. *Comput. Appl. Biosci.*, **13**, 477–478.

Ning,Z., Cox,A.J. and Mullikin,J.C. (2001) SSAHA: a fast search method for large DNA databases. *Genome Res.*, **11**, 1725–1729.

Pearson,W.R. and Lipman,D.J. (1998) Improved tools for biological sequence comparison. *Proc. Natl Acad. Sci. USA*, **85**, 2444–2448.

Rognes,T. and Seeberg,E. (1998) SALSA: improved protein database searching by a new algorithm for assembly of sequence fragments into gapped alignments. *Bioinformatics*, **14**, 839–845.

Skiena,S.S. (1997) *The Algorithm Design Manual*. Springer, New York.

Venter,J.C., Adams,M.D., Myers,E.W., Li,P.W., Mural,R.J., Sutton,G.G., Smith,H.O., Yandell,M., Evans,C.A., Holt,R.A. *et al.* (2001) The sequence of the human genome. *Science*, **291**, 1304–1351.

Wheelen,S.J., Church,D.M. and Ostell,J.M. (2001) Spidey: a tool for mRNA-to-genomic alignments. *Genome Res.*, **11**, 1952–1957.

Zhang,Z., Schwartz,S., Wagner,L. and Miller,W. (2000) A greedy algorithm for aligning DNA sequences. *J. Comput. Biol.*, **7**, 203–214.