Evaluation of Unique Sequences on the European Data Grid

Ken-ichi Kurata[1], Christian Saguez[1], Gerard Dine[1], Hiroshi Nakamura[2] and Vincent Breton[3]

[1]Laboratoire de Mathematiques Appliquees aux Systemes,
Ecole Centrale Paris,
Grande voie des vignes, Chatenay-Malabry 92295, France
[2]Research Center for Advanced Science and Technology,
The University of Tokyo,
4-6-1 Komaba, Meguro-ku, Tokyo 153-8904, Japan
[3]Laboratoire de Physique Corpusculaire de Clermont-Ferrand,
Centre National de la Recherche Scientifique,
24 avenue des Landais, Aubiere 63177, France
Email: Ken-ichi.Kurata@mas.ecp.fr

Abstract

Thanks to the development of genetic engineering, various kinds of genomic information are being unveiled. Hence, now, it becomes feasible to study in molecular biology by analyzing the entire genomic information. On the other hand, the quantity of the genomic information stocked in database is increasing day after day. In order to process the whole information, we have to develop an effective method to deal with lots of data. It is indispensable not only to make an effective and rapid algorithm but also to use high-speed computer resource so as to analyze the biological information. For this purpose, as one of the most promised computing environments, the grid computing architecture has appeared recently. The European Data Grid (EDG) is one of the grid computing environments (Segal 2000).

In this paper, at first, we propose an effective sequence analysis method to find out and evaluate unique sequences for primer design by extending the previously proposed method (Kurata 2002). Next, we describe how to implement this method upon the European Data Grid.

Keywords: unique sequence, primer design, the European Data Grid

1 Introduction

The genomic sequences are being unveiled on a lot of target organisms. The development of genetic engineering accelerates this process. Hence, it has been indispensable to analyze the whole genomic information in molecular biology. A great amount of gene sequences are discovered by the new technology and these are being stocked in public and private databases day after day. Thus, we have been able to support molecular biological experiments by analyzing them. One of the traditional biological methods is Polymerase Chain Reaction (PCR), which is used broadly and usually in molecular biology, such as gene therapy, gene diagnosis, DNA sequencing and gene expression pattern observation. In order to do successful experiments in PCR, it is important to design target-specific sequences used as primers from the genomic information.

On the other hand, it is said that the quantity of stocked genomic information should double every 8 months. The more the quantity of genetic information augments, the more computation power is required. Thus we have to develop new methods of analyzing such information efficiently and rapidly. One of the solutions is to implement and process the algorithm in distributed computing environment. Now, thanks to the development of the infrastructure of high-speed networks, it is becoming feasible to deploy the distributed computing environment on the Internet. One of the projects that realize this type of computing environment is called the European Data Grid project (Segal 2000). This project provides the users with the distributed computing environment to deal with the problems hard to resolve. Providing the infrastructure and tools that make large-scale, secure resource sharing possible and straightforward is the Grid's raison d'etre (Foster 2001, Foster 2002).

We have proposed a method to find genuinely unique sequences on target genes (Kurata 2000). In order to design the primers that produce a specific product, the specificity of a primer itself is taken into consideration. In addition, we have proposed the method that ensures the specificity of a pair of primers, namely, the uniqueness of its product length in PCR (Kurata 2002). In these previous works, the length of the unique sequences is used as the criterion for selecting optimal sequences. As it were, the unique sequence whose length is as short as possible is selected. However, it is possible that the unique and short sequence consists of partial sequences existing frequently on the genome. Even if this type of sequence is used as a primer, the sequence is surely unique on its target genome but its partial sequences can frequently hybridize the other sequences. This method can not properly deal with such sequences.

In this paper, we propose a new way to evaluate the specificity of sequences. We select a sequence candidate by paying attention to the local frequency of occurrence of its partial sequences. Furthermore, we describe an implementation of this method onto the European Data Grid and show the result of computing experiments upon it.

2 Target specific primer in PCR

In this section, we discuss the condition of the target specific primer. At first, specificity of sequence itself as a primer is discussed. Next, specificity of sequence as a pair of primers is shown.

2.1 Specificity of sequence as a primer itself

In this part, firstly, the condition of proper primer sequences is shown. Secondly, the condition of unique

sequences is described. Finally, the condition of the target specific primer is summarized.

A primer must strictly hybridize with a target gene and avoid hybridizing with non-target genes. The reaction of hybridization in the 3'-end region influences most on the total PCR or RT (Griffais 1991). On the other hand, the start of elongation reaction in PCR is relatively insensitive to the hybridization reaction in the 5'-end region.

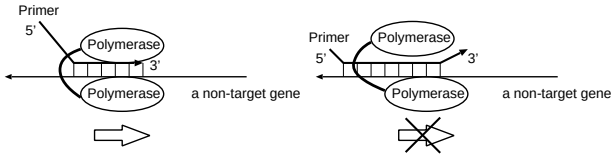

Figure 1: The illustration on the way of commencement in elongation reaction. On the left, a mismatch occurs in the 5'-end region of the sequence. On the right, a mismatch occurs in the 3'-end region of the sequence.

Figure 1 illustrates how the elongation reaction proceeds. On the left, when the partial sequence in the 3'-end region of the primer hybridizes with a non-target gene, the elongation reaction begins without the hybridization reaction in the 5'-end region. On the contrary, as shown on the right, if the sequence does not hybridize in the 3'-end region, it is difficult for the polymerase to start the elongation reaction. When a mismatch against non-targets occurs on the sequence near the 3'-end of the primer, the elongation reaction hardly begins. Only if the sequence of the 3'-end region of the primer is enough specific, the correct elongation reaction can proceed efficiently. The efficiency of hybridization is mainly influenced by the consecutive region of sequence without mismatch (Allawi 1997). Therefore, the most important factor to drive elongation reaction accurately and efficiently is the uniqueness of the sequence in the 3'-end region. Hence, we must select the sequence that has a unique and short sequence in the 3'-end region as an optimal primer.

We have proposed how to calculate the minimum length of a unique consecutive sequence (USL) in our previous work (Hosaka 2001). USL means the minimum length of a unique sequence. If the length of a partial sequence of a gene is beyond the value of its USL, the sequence becomes unique on the whole target genome. By means of this method, some short sequences that exist only once on the entire target genome can be found. In other words, the uniqueness of the sequences on the target genome is guaranteed by this method, unlike the frequency of occurrence method (Kurata 2000). If the sequence existing more than twice on the entire target genome is used as a primer, the specific band is never produced.

The sequence of a target specific primer should not exist more than twice on the target genomic sequence. If a sequence frequently occurring on the target genome is used as a primer, a lot of unexpected products are amplified. Namely, the important signal could be subdued by much noise. Thus, as for the target specific primer, we claim the following characteristics:

1. The 3'-end sequence of the primer must exist only once on the whole genome, namely it must be unique.

2. The unique sequence must be as short as possible.

2.2 Specificity of sequence as a pair of primers

In this part, at first, we discuss the condition of PCR products. Next, the condition of a sequence-pair amplifying a unique product in PCR is described.

In PCR, in order to amplify the partial sequence that we desire, we can use two oligo sequences as a pair of primers. The partial sequence sandwiched between one primer and the other is amplified.

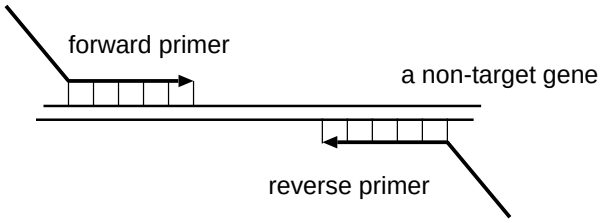

Figure 2: The illustration on how PCR works. Both primers hybridize with a non-target gene.

Figure 2 and 3 illustrate how PCR works. In Figure 2, the consecutive sequence in the 3'-end region of a forward primer is matching the partial sequence of a non-target. That of the reverse primer is also matching the partial sequence of the non-target. On this condition, both primers possibly hybridize with the non-target gene and the elongation reaction probably begins on both sequences. After n cycles of PCR, the quantity of its wrong product grows by the factor of 2^n.

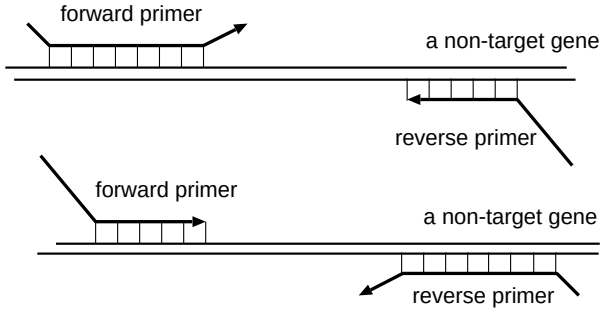

Figure 3: The illustration on how PCR works. one of the primers hybridizes with a non-target gene.

To the contrary, as shown in Figure 3, when either forward primer or reverse one is specific, it is difficult for the polymerase to accomplish the reaction. When one primer is not matching the non-target, even if the other primer is matching, the quantity of its wrong product is proportional to at most $2n$ after n-cycle PCR. Thus, in PCR, only when the mishybiridization reaction on both primers takes place, the wrong product is amplified. Namely, we can say that the reaction induced by 2 sequences contributes to the total reaction in PCR. In other words, the elongation reaction in PCR is influenced by the specificity of 2 sequences. Hence, we must select the sequence that has a unique and short sequence on either edge as an optimal sequence.

Besides that, the amplified product can be distinguished in length. Even if several regions of target or non-target genes are amplified at a time, it is still possible to discern their existence by observing their product length. In other words, we can distinguish the specific product from among the others by looking at their length by means of electrophoresis. Inversely speaking, it is necessary to circumvent the possibility

of wrong PCR products whose length is the exactly same as that of the target. Therefore, we must take into consideration the uniqueness of product length. In short, the following is required.

1. The edge sequences of a product must be unique.
2. The uniqueness of product length must be taken into account.
3. These unique sequences must be as short as possible.

3 Local Frequency of Occurrence (LFO)

In this section, we propose a method by which sequence candidates are selected. This method is applied for selecting a sequence as a primer itself. In this part, we discuss a problem of some primer design methods existing now. As one of the solutions, we introduce local frequency of occurrence and apply it for primer sequence selection.

First of all, we glance at problems on the sequence design method that exists up to now. The method based on frequency of occurrence is shown. Secondly, the way of finding genuinely unique sequences on a target genome is discussed. Thirdly, we discuss the problem brought about by these means. Finally, we describe how to circumvent this problem and propose a sequence evaluation method.

3.1 Frequency of Occurrence method (FO)

The following strategy is one of the classical but popular methods of designing primers. At first, one finds short sequences whose frequency of occurrence is low, namely, which occur rarely on the whole genome. Next, an adequate primer sequence composed of such short sequences is selected. This method is called frequency of occurrence method (FO). In FO, however, an grave discrepancy exists. It is not sure whether the sequence composed of low frequent sequences uniquely exists on the whole genome or not. In other words, in the above method, the uniqueness of the primer is not guaranteed.

Let us consider the short genome 5'-tttcgtttgcgctaactagcgttt-3'. When the window size of FO is 1, nucleotide 'a', 'c', 'g' and 't' happen 3, 5, 5 and 11 times on the genome, respectively. Now, we make a primer whose length is 3 on the genome by taking into account the value of FO. The value of FO is evaluated as follows. For example, 'act' consists of 'a', 'c' and 't', thus, the value of FO of 'atc' is 3+5+11 = 19. In the same manner, the value of FO of 'gcg' is 5+5+5 = 15. At this time, the problem of the evaluation based on FO has occurred. Looking at the value of FO of each sequence, the value of 'act' is more than that of 'gcg'. If the specificity of the sequences is judged by the value of FO, it seems that the latter one, 'gcg', is superior to the former one, 'act'. However, 'gcg' occurs twice on the genome, namely, 'gcg' is not unique. If this sequence is used in PCR, unexpected genes are possibly amplified. On the other hand, 'act' is unique on the genome. In short, the uniqueness of the sequence can not be ensured on the basis of the value of FO. Even if the larger window size is used, the same problem happens. Furthermore, in general, the size of FO table is proportional to that of hash table. Hence, the larger the window size is, the more difficulties in memory we have to deal with.

3.2 Finding unique sequences

Therefore, we must conceive a method to ensure the uniqueness of primer. Now, we can take an effective strategy so as to find unique sequences on the whole genome. As a method to discover a unique sequence on the genome, there are some algorithms. One is the method by using suffix tree (or suffix array) (Fugen 2001), the other is the method by using radix sort, which we proposed (Kurata 2002, Hosaka 2001). In these methods, we can find genuinely unique sequences on the entire genome. However, the problem on how to evaluate the specificity of unique sequences is left to be solved. In our previous method, we make use of the length of unique sequences as the criterion of selection. The unique sequence whose length is as short as possible is selected. In this method, however, we can not evaluate the specificity of the elements composing the unique sequence. That is to say, even if a unique sequence is found, it is possible that a part of the sequence might match many other parts of the entire genome. Moreover, even if we use the value of FO as the criterion, this problem can not be resolved. Let us consider again the genome 5'-tttcgtttgcgctaactagcgttt-3'. There are sequence 'acta' and 'tcg' on the genome. Both of them are unique. If these sequences are compared in length, 'acta' is longer than 'tcg'. Hence, it seems that the sequence 'tcg' is superior to the sequence 'acta' based on the criterion of length. Next, we calculate the value of FO for each sequence. The value of FO of 'tcg' is 11+5+5 = 21. On the other hand, the value of FO of 'acta' is 3+5+11+3 = 22. If the criterion based on the value of FO is applied to these unique sequences, it seems that the sequence 'tcg' is more suitable for a primer than the sequence 'acta'. However, in PCR, the hybridization reaction in the region near the 3'-end of primer is most important. The partial sequence 'cg', which exists in the 3'-end region of 'tcg', occurs 3 times on the genome. But the partial sequence 'ta', which exists in the 3'-end region of 'acta', occurs twice. Thus, in the 3'-end region, it is said that the mis-hybridization reaction of 'tcg' happens more frequently than that of 'acta'. This problem is caused by evaluating the specificity on the basis of the calculated values of FO on the whole genome.

3.3 Local Frequency of Occurrence method (LFO)

In order to avoid the above risk, we propose a method to calculate the value of Local Frequency of Occurrence (LFO) and use it as the criterion. The value of LFO of a sequence represents the value of FO calculated on the local sequence, not as a whole. In the following, we explain this method. In hybridization reaction, the length of the matching sequence is the most important factor. On the other hand, the short sequence frequently existing on the target genome does not befit a partial sequence of a primer.

Figure 4 illustrates on the length of unique sequences and its LFO. Now, let us hash the genome 5'-tttcgtttgcgctaactagcgttt-3' on the basis of each nucleotide. When it comes to the way of hashing all the sequences, we discussed it in our previous work (Kurata 2002) and we explain it in the following section.

As shown on the left of this figure, there are 5 sequences having the hash-key sequence 'g' on the entire genome. In the same manner, as shown on the right, we can find 3 sequences having the other hash-key sequence 'a'. When we select proper sequences on the basis of the length of unique sequence as we proposed formerly, the partial sequence 'tcg' on the genome is more suitable for a primer than 'acta', be-

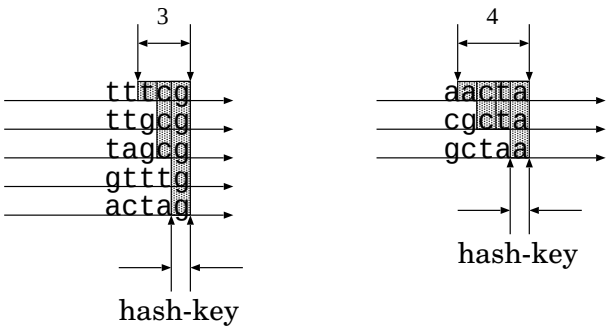

Figure 4: The illustration on the length of unique sequences and its local frequency of occurrence. On the left, 5 sequences having the same hash-key 'g' exist. The length of the unique sequence on the most upper part is 3. On the right, 3 sequences having the same hash-key 'a' exist. The length of the unique sequence on the most upper part is 4.

cause the length of 'tcg' is shorter than that of 'acta' as a unique sequence. However, when paying attention to each hash-key, we find that the hash-key 'g' appears 5 times, on the other hand, the hash-key 'a' appears 3 times. Even if the sequence 'tcg' is used as a primer, the sequence itself is certainly unique. But judged by the hash-key, this sequence has more probability of hybridization in the 3'-end region than the sequence 'acta' has. In short, we would like to distinguish these sequences on the basis of the local frequency of occurrence.

In order to resolve this problem, we propose the following evaluation method. The length of hash-key H is defined as $|H|$. The number of the sequence existing on the target genome followed by the hash-key sequence H at the 3'-end, whose total length is $l + |H|$, is assigned to $|sequence(l; H)|$. Namely, $sequence(0; H)$ represents the hash-key H. For instance, when the hash-key sequence is assigned to 'g', atg(0;'g') = 'g', atg(1;'g') = 'tg' and atg(2;'g') = 'atg'.

In other words, l represents the position from the 5'-end of the hash-key. Here, we define the following function.

$$f_{lfo}(sequence) = \sum_{i=0}^{ul} \log |sequence(i; H)|$$

ul stands for the unique length of this sequence. Namely, when the length of this sequence is beyond this value, it becomes unique on the target genome.

This function is easily implemented into our proposed sorting method (Kurata 2002). As for the implementation, we show it in the following section. Here, we evaluate the sequences discussed above by using this function. The value calculated for the partial sequence 'tcg' on the genome is $f_{lfo}(tcg) = |g| + |cg| + |tcg| = \log 5 + \log 3 + \log 1 = \log 15$. The value calculated for the partial sequence 'acta' is $f_{lfo}(acta) = |a| + |ta| + |cta| + |acta| = \log 3 + \log 2 + \log 2 + \log 1 = \log 12$. Therefore, if evaluated by the value based on the above function, the value of 'acta' is less than that of 'tcg', that is to say, the latter one is evaluated as the more suitable sequence.

4 Algorithm

The goal is to find a unique sequence as short as possible, which exists only once on the target genes. Moreover, we require that the value of LFO of the unique sequence should be small. It is computationally demanding to naively compare all partial sequences of the target genes with one another. Hence, we take the following strategy to find such sequences rapidly. First of all, the total task is decomposed into small jobs. The number of the combinations of sequences to be compared with each other is reduced by hashing all the genomic information. Each job is allotted to one of Computing Elements (CEs) on the Data Grid. Next, an algorithm like radix sort processes the comparison among sequences on each computing element. Finally, all the results are gathered and candidate sequences are selected.

Our method is composed of 4 steps, described as follows: (1) make a Look Up Table (LUT) from the whole genomic information; (2) arrange the calculation on the European Data Grid (EDG); (3) calculate unique sequence length (USL) and local frequency of occurrence (LFO) by using the algorithm like radix sort; (4) select the candidate sequences for a primer.

4.1 Construction of LUT

In this part, we describe the way of constructing Look Up Table(LUT). Firstly, we explain how to make a LUT for primer itself. Secondly, we describe how to make a LUT for primer-pair.

4.1.1 LUT for primer itself

First of all, a LUT is made by using the information of all the target genes.

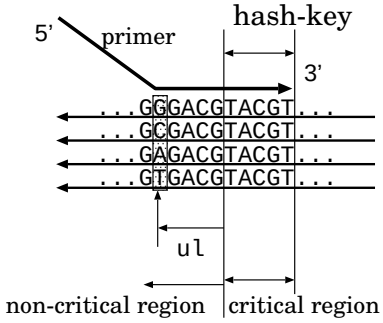

Figure 5: This illustration shows how to assign a hash-key sequence on target gene sequences. Each partial sequence on the targets is allotted to a hash-key sequence. These hash-key sequences are used as the 3'-end sequence of a primer. The matching length from the 5'-end of the hash-key sequence is given as unique length (ul).

The hybridization of the consecutive sequence in the 3'-end region is the most important in the total reaction. In other words, the hybridization reaction in the 3'-end region is more critical than the reaction in the 5'-end region. Therefore, as shown in Figure 5, we assign each hash-key sequence to the 3'-end sequence of a primer. The matching length from the 5'-end of the hash-key sequence is given as unique length (ul). Now, an outline is given as follows.

Suppose that there is an organism that has l genes. The gene of the target organism is described as $g_i, i = 1, \ldots l$. The length of each gene is given as $|g_i| = m_i$ and the nucleotide of position j from the 5'-end of gene i is specified as $g_i(j)$, $g_i(j) \in \{a, t, g, c\}$. Now, a LUT for all the positions of all the genes, $j = 1, \ldots, m_i, i = 1, \ldots, l$, is made. Each subsequence $g_i(j) \ldots g_I(j + h - 1)$ of position j from the 5'-end of gene i is used as the hash-key sequence for $LUT(i, j)$. Now, the hash-key sequence is assigned to the 3'-end region of primer, as shown in Figure 5. The length of the hash-key sequence is h, so that the size of the 3'-end region is h. The LUT includes the pointer to the

next position at which the same hash-key sequence appears. All the positions that have the same hash-key sequence are quickly found only by searching the LUT. The size of the LUT is proportional to that of the genome.

An example of the algorithm in detail is shown as follows. Let us assume that there is an organism that has a very short genome "5'-ttaacaagtcaagtcaagacaa-3'". Here, the size of hash-key is appointed 2, namely, $h = 2$.

Position:	0	1	2	3	4	5	6	7	8	9	10
Sequence:	t	t	a	a	c	a	a	g	t	c	a
LUT :	-1	-1	5	18	9	10	11	12	13	14	15

Position:	11	12	13	14	15	16	17	18	19	20	21
Sequence:	a	g	t	c	a	a	g	a	c	a	a
LUT :	16	-1	-1	19	20	-1	-1	-1	-1	-1	-1

Table 1: An example of the LUT for primer itself

The LUT is composed as shown in Table 1. '-1' means terminal signal in LUT. If this signal occurs in LUT, after that, no sequence including the same hash-key sequence exists. Now, let us find the hash-key sequence, 'aa'. At first, this sequence appears at position 2. Next, we can find it at position 5 on the genomic sequence only by looking at the LUT. The pointer to the next position is given at the position 2 of the LUT. Similarly, all the positions, 10, 15 and 20, where 'aa' appears, are found by using the LUT.

4.1.2 LUT for pair of primers

Here, we explain how a LUT for primer-pairs is made by using the whole genomic information of the target organism. As described above, the hybridization of the consecutive sequence in the 3'-end region of a primer is most important in the total PCR. Moreover, the uniqueness of the product length must be ensured.

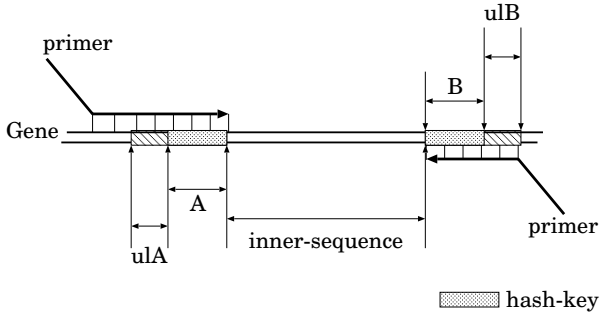

Figure 6: This illustration shows how to assign a hash-key sequence on target gene sequences. The sequence composed of both edge sequences of all partial sequences on the targets are allotted to a hash-key sequence. Namely, The concatenated sequence A + B is used as the hash-key sequence. These hash-key sequences are used as the 3'-end sequence of each primer. The matching length from the hash-key sequence in each direction is given as the unique length of A (ulA) or the unique length of B (ulB), respectively. The size of ulA + ulB is used as the criterion similar to the unique length discussed in the previous section.

Therefore, as shown in Figure 6, we make a hash-key sequence by concatenating one edge sequence with the other one on a PCR product when making LUT. Namely, we use the sequence A + B as a hash-key. Now, we call the sequence intercalated between A and B "inner-sequence". As shown in this figure, the 3'-end sequence of the forward primer is sequence A and that of the reverse primer is sequence B. On one hand, we can select a large value as the length of inner-sequence in order to amplify a sequence for cloning and sequencing. On the other hand, we can select a small value in order to amplify a sequence for gene expression pattern observation in PCR. Besides that, when the size of inner-sequence is assigned to 0, we can evaluate the specificity of the probes for Ligase Chain Reaction (LCR).

The LUT is made as follows. Suppose that there is an organism that has l genes. The gene of the target organism is described as $g_i, i = 1, \ldots l$. The length of each gene is given as $|g_i| = m_i$ and the nucleotide of position j from the 5'-end of gene i is specified as $g_i(j)$, $g_i(j) \in \{$ a,t,g,c $\}$. Now, LUT for all the positions of all the genes, $j = 1, \ldots, m_i, i = 1, \ldots, l$, is made. Each subsequence $g_i(j) \ldots g_i(j+h-1) g_i(j+h+s) \ldots g_i(j+2h+s-1)$ of position j from the 5'-end of gene i is used as the hash-key sequence for $LUT(i, j)$. The length of the hash-key sequence is assigned to $2h$, that is, the length of A + B is $2h$. The length of the inner-sequence is s. LUT includes a pointer to the next position at which the same hash-key sequence appears. All the positions that have the same hash-key sequence are quickly found only by searching the LUT. The size of the LUT is proportional to that of the target.

An example of the algorithm in detail is shown as follows. Let us assume that there is the organism that has a very short genome "5'-tgaatgcgaaccccaacgcgaataccaacgctaatat-3'". Here, the size of hash-key is appointed 2, $h = 2$. Namely, each of the 3'-end sequence of the primer is 2. The size of inner-sequence is 4, namely, $s = 4$.

Position:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Sequence:	t	g	a	a	t	g	c	g	a	a	c	c	c	c	a	a	c	g
LUT :	-1	-1	9	22	-1	-1	-1	-1	15	-1	-1	-1	-1	-1	21	-1	29	-1

Position:	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
Sequence:	c	g	a	a	t	a	c	c	a	a	c	g	c	t	a	a	t	a
LUT :	-1	-1	27	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

Table 2: An example of the LUT for pair of primers

The LUT is composed as shown in Table 2. '-1' means terminal signal in LUT. If this signal occurs in LUT, after that, no sequence including the same hash-key sequence exists. Now, let us find the hash-key sequence, `'aaaa'`, namely sequence `'aa----aa'`. Here, `'----'` means an arbitrary sequence whose length is 4. At first, this sequence appears at position 3. Next, we can find it at position 9 on the genomic sequence only by looking at the LUT. The pointer to the next position is given at position 3 of the LUT. Similarly, all the positions, 15, 21 and 27, where `'aa----aa'` appears, are found by using the LUT.

4.2 Deployment on the European Data Grid

The goal of the European Data Grid Project (Segal 2000) is the development of a novel environment to support globally distributed scientific exploration involving multi-PetaByte datasets. The project is designing and developing middle-ware solutions and testbeds capable of scaling to handle PetaBytes of distributed data, tens of thousands of resources (processors, disks, etc.), and thousands of simultaneous users. DataGrid biomedical work package gathers biologists, computer scientists, physicians and physicists around the common goal of deploying a biomedical grid (Breton 2001) .

We describe the structure of the European Data Grid (EDG) here. The EDG is mainly composed of 4 computing elements: User Interface (UI), Resource

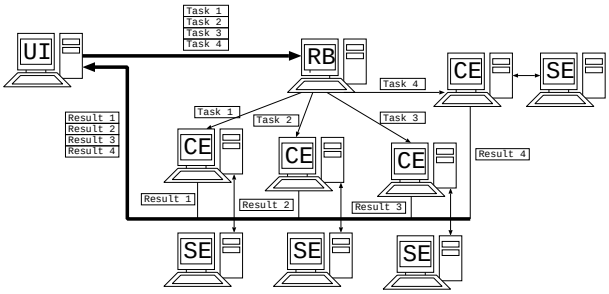

Figure 7: The illustration on the working flow upon the Data Grid. UI, RB, CE and SE mean User Interface, Resource Broker, Computing Element and Storage Element, respectively.

Broker(RB), Computer Element(CE) and Storage Element(SE). At first, a user belonging to the EDG can contact with all the machine through a UI machine in place. On the UI machine, all the necessary commands to issue jobs upon the EDG are furnished. In other words, all the users hoping to process their jobs on the EDG have to use a UI machine as an entrance for this environment. Next, all the jobs issued with the UI are sent to a RB machine. The RB machine having received the commands from the UI seeks for available computer resources, that is, CEs, and schedules all the jobs. When the RB machine finds an available CE, it sends one of the jobs to this CE. The CE machine processes the received jobs. If the CE machine is ordered to use some databases, it seeks for the SE having the requested data and orders them. Finally, the termination of the calculation is reported to the UI through the RB and the user can gather all the results on the UI machine.

In this research, at first, we issue all the jobs through a UI, as shown in Figure 7. Secondly, the whole task is decomposed into some smaller tasks on the basis of the hash-key discussed above. The machine contacts a RB and each task is distributed to one of the available CEs. In each CE, the following is processed.

4.3 Calculation of Unique Sequence Length (USL) and Local Frequency of Occurrence (LFO)

Next, unique sequence length is calculated at all the positions of the target genes. In addition, as mentioned above, the value of local frequency of occurrence of each unique sequence is also calculated. We have already proposed the method to find unique sequences on the whole genome (Kurata 2002). Here, we improve our previously proposed method to be able to calculate the value of LFO. In this section, at first, the method of calculating USL and LFO for primer itself is shown. Next, the calculation method for a pair of primers is discussed.

4.3.1 Calculation of USL and LFO for primer itself

Here, the problem is how to calculate the value of USL and LFO. The method that naively compares all the sequences with each other spends a great deal of computation time. Thus, we propose an effective method to calculate the value of LFO by using an algorithm based on radix sort.

Now, as shown in Figure 5, let us consider subsequence $g_{i_k}(j_k) \ldots g_{i_k}(j_k + h - 1)$ of gene g_{i_k} as a hash-key sequence. The hash-key sequence is assigned to the sequence of the 3'-end region of primer. Suppose that the hash-key sequence appears n times on the entire genome as subsequence $H = g_{i_k}(j_k) \ldots g_{i_k}(j_k + h - 1)$ of gene g_{i_k}, $k = \{1, \ldots, n\}$.

Here, the following comparison is done for every hash-key. In the following, ':=' represents substitution. lfo represents the value of LFO. Here, unique length (ul) is the matching length from the 5'-end of the hash-key. $N = \{1, \ldots, n\}$ is the set of all the positions where the same hash-key occurs. R_r is a subset of set N, and $r = \{a, c, g, t\}$. r signifies the name of sorting buckets. Here, before sorting, the following substitution is executed. $s_k(x) := g_{i_k}(j_k - x), k = \{1, \ldots n\}, x = \{1, \ldots Th\}$. Th means the threshold where the comparison stops.

Initially, $ul := 0, lfo = 0$. The following sorting function is executed.

SORT(N, ul, lfo)

1. $ul := ul + 1$.
2. If the size of N is 1, then subsequence $s_k(ul) \ldots s_k(1)H, k \in N$ is a unique sequence, whose length is $ul + h$ and whose value of LFO is lfo.
3. $lfo := lfo + \log(|N|)$. $|N|$ represents the size of N.
4. Initially, $R_r := \phi$ for all the elements of $r = \{a, c, g, t\}$, and the following is done for all the elements of N, $k \in N, r = \{a, c, g, t\}$.
   (a) If $s_k(ul)$ is nucleotide 'a', then $R_a := R_a \oplus k$.
   (b) If $s_k(ul)$ is nucleotide 'c', then $R_c := R_c \oplus k$.
   (c) If $s_k(ul)$ is nucleotide 'g', then $R_g := R_g \oplus k$.
   (d) If $s_k(ul)$ is nucleotide 't', then $R_t := R_t \oplus k$.
5. If the size of R_a is more than 1, then call the function, **SORT**(R_a, ul, lfo) .
6. If the size of R_c is more than 1, then call the function, **SORT**(R_c, ul, lfo) .
7. If the size of R_g is more than 1, then call the function, **SORT**(R_g, ul, lfo) .
8. If the size of R_t is more than 1, then call the function, **SORT**(R_t, ul, lfo) .

The minimum length of a unique sequence and its value of local frequency of occurrence are calculated by the above method.

For example, here, we consider the genome 5'-tttcgtttgcgctaactagcgttt-3'. Now, the size of hash-key is 1. Here, let us make a comparison among the partial sequences including hash-key 'g'. 5 partial sequences including 'g' are found on this genome by using its LUT. Now, the comparison is executed using the proposed algorithm as follows. 5 partial sequences including 'g' are shown in Table 3.

1	tttcg
	tttcgtttg
	tttcgtttgcg
	tttcgtttgcgctaactag
	tttcgtttgcgctaactagcg

Table 3: An example of the proposed sorting algorithm: 5 partial sequences including 'g'

Since there are 5 sequences, the initial value of LFO of these sequences is $\log(5) = 1.61$. At first, these sequences are sorted on the basis of their character left to the hash-key sequence 'g' as shown in Table 4.

	LFO	sequence
1	1.61	tttcgtt-t-g
2	1.61	tttcgtttgcgctaact-a-g
3	1.61	ttt-c-g
		tttcgtttg-c-g
		tttcgtttgcgctaactag-c-g

Table 4: An example of the proposed sorting algorithm: the sequences sorted on the basis of their character left to the hash-key sequence 'g'

The sequences that have character 't' are sorted in row 1. The sequences that have character 'a' are sorted in row 2. Now, there is only one sequence in row 1 and row 2. Hence, the sequence 'tg' and 'ag' are unique. Their final value of LFO is 1.61. The sequences that have character 'c' left to the hash-key are sorted in the bottom row. Since there are 3 sequences in this row, $\log(3) = 1.10$ is added to the value of LFO of these sequences. Next, the sequences are sorted in row 3 on the basis of the second character from the hash-key 'g'. The result is shown in Table 5.

	LFO	sequence
1	1.61	tttcgtt-t-g
2	1.61	tttcgtttgcgctaact-a-g
3	2.71	tt-t-cg
4	2.71	tttcgttt-g-cg
		tttcgtttgcgctaacta-g-cg

Table 5: An example of the proposed sorting algorithm: the sequences sorted on the basis of the second character from the hash-key sequence 'g'

The sequences that have character 't' are sorted in row 3. Now, there is only one sequence in row 3. Hence, the sequence 'tcg' is unique. Its final value of LFO is 2.71. The sequences that have character 'g' are sorted in the bottom row. Since there are 2 sequences in this row, $\log(2) = 0.69$ is added to the value of LFO of these sequences.

Finally, These sequences are sorted in the same manner in each row until all the sequences become unique or ul reaches the given threshold. The result is shown Table 6

	USL	LFO	sequence
1	2	1.61	tttcgtt-t-g
2	2	1.61	tttcgtttgcgctaact-a-g
3	3	2.71	tt-t-cg
4	4	3.40	tttcgtt-t-gcg
5	4	3.40	tttcgtttgcgctaact-a-gcg

Table 6: The result of the example of the proposed sorting algorithm. USL means the length of unique sequence. LFO means the value of local frequency of occurrence.

By means of this, we can calculate the length of all the unique sequences on the whole genome. Furthermore, we can calculate the value of LFO simultaneously.

In the same way, we compare partial sequences with each other according to each hash-key sequence. The final results are shown in Table 7.

When looking at Table 7, we can find out that the unique sequences whose length is short do not always have small value in LFO. The length of the unique sequence 'tgc' is the same as that of 'gct' Hence, on the basis of the length of these unique sequence, we

	USL	LFO	sequence
	4	2.484910	gcta
	2	1.098610	aa
	4	2.484910	acta
	2	1.609440	tc
	3	2.708050	tgc
	3	2.708050	cgc
	2	1.609440	ac
	3	2.708050	agc
	3	2.708050	tcg
	2	1.609440	tg
	4	3.401200	tgcg
	2	1.609440	ag
	4	3.401200	agcg
	4	3.784190	tcgt
	5	5.575950	tcgtt
	6	6.674560	tcgttt
	3	3.091040	gct
	3	3.091040	act
	4	3.784190	gcgt
	5	5.575950	gcgtt
	6	6.674560	gcgttt

Table 7: The final result of the example of the proposed sorting algorithm.

can not distinguish the specificity of 'tgc' from that of 'gct'. Moreover, if the value of FO is calculated for each sequence on condition that the window size of FO should be 1, similarly, these unique sequences can not be classified. However, the value of LFO of the sequence 'tgc' is smaller than that of 'gct'. Hence, 'tgc' is superior to 'gct' in our criterion. Furthermore, the value of LFO of the sequence 'acta' is smaller than that of 'tgc'. If the specificity of the sequences is evaluated by their length, 'acta' should be inferior to 'tgc' and 'gct'. However, 'acta' is superior to 'tgc' and 'gct' in our criterion.

4.3.2 Calculation of USL and LFO for primer-pair

Here, the method to calculate USL and LFO for primer-pair is shown.

Now, as shown in Figure 6, let us consider subsequence $g_{i_k}(j)\dots g_{i_k}(j+h-1)g_{i_k}(j+h+s)\dots g_{i_k}(j+2h+s-1)$ of gene g_{i_k} as a hash-key sequence. The hash-key sequence is assigned to the sequence of the 3'-end region of primer. Suppose that a hash-key sequence, H, appears n times on the genome as subsequence $g_{i_k}(j)\dots g_{i_k}(j+h-1)g_{i_k}(j+h+s)\dots g_{i_k}(j+2h+s-1)$ of gene g_{i_k}, $k = \{1,\dots,n\}$.

Here, the following is done for every hash-key. In the following, ':=' represents substitution.

$$\begin{aligned} s_k(2x-1) &:= g_{i_k}(j_k - x) \\ s_k(2x) &:= g_{i_k}(j_k + 2h + s - 1 + x) \\ & \quad k = \{1,\dots n\} \\ & \quad x = \{1,\dots Th\} \end{aligned}$$

Th means the threshold where the comparison stops.

Initially, $ul := 0, lfo = 0$. lfo represents the value of LFO. Here, unique length (ul) stands for the total matching length, ulA + ulB, as shown in Figure 6. In other words, ulA represents the matching length from the 5'-end of the hash-key and ulB represents the matching length from the 3'-end of the hash-key. $N = \{1,\dots,n\}$ is the set of all the positions where the same hash-key occurs. R_r is a subset of set N, and $r = \{a, c, g, t\}$. r signifies the name of sorting

buckets. Now, the same sorting function as described in the preceding section, **SORT**(N, ul, lfo) , is called.

In regard with the sorting method to design a pair of primers, we have already discussed it in our recent work (Kurata 2002).

4.4 Sequence selection for primer

Finally, we select the sequences that have the small value of LFO. A partial sequence whose value of LFO is small befits a primer candidate. This operation is executed on the entire target genome.

5 Implementation

In this section, we describe the program executing the proposed method and its implementation on the EDG. At first, we describe the computing environment used in this work. Next, the working flow of the program is shown.

5.1 Calculation environment

The environment of the European Data Grid was installed and configured with LCFG server box (Iven 2002). LCFG is one of the standard installation systems of the Data Grid environment. Once the LCFG server is installed, it automatically provides all the client machines with all the provisions to bootstrap. Moreover, all the client machines are automatically maintained and configured by the LCFG server. In this research, the Data Grid environment based on Globus 2 beta was installed through the LCFG server. All the main elements, such as User Interface (UI), Computing Element (CE), Storage Element (SE) and Working Node (WN) are installed and configured with this system. For the moment, Redhat Linux 6.2 is supposed to be installed into all the machines used on the Data Grid environment.

As of the sequence analysis program for selecting the sequences, it was written in C++. This program consists of 2 main modules. The first module works in order to hash the entire target genomic information and classify it into some smaller sets of partial sequences. The second module processes one of the sets derived from the entire target genomic information by the first module. It sorts all the input data and outputs the length of the unique sequences together with the value of LFO of them.

In order to implement this program onto the Data Grid environment, we made use of Job Description Language (JDL). All the calculation was processed in the environment of Figure 8.

First of all, we issued all the commands from the UI machine of Clermont-Ferrand located in France. The calculation was divided into some jobs. Every job, together with needed files, was sent to the Resource Broker (RB) machine located in the European Organization for Nuclear Research (CERN) in Switzerland. This machine scheduled and dispersed all the jobs to available CEs on the Data Grid. Each CE processed the job received from the RB and returned the result to the UI. In this experiment, some CE machines, which are located in CERN, in the Rutherford Appleton Laboratory (RAL) in England, in the Instituto Nazionale di Fisica Nucleare in Italy, in the National Institute for Nuclear Physics and High Energy Physics (NIKHEF) in Netherlands and in Centre de Calcul de Institut National de Physique Nucleaire et de Physique des Particules (CC-IN2P3) in France, were used. In other words, all the jobs are allocated by the RB to these CEs.

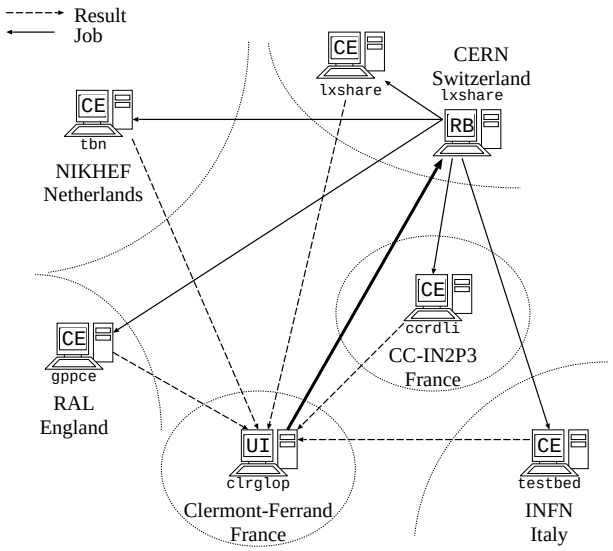

Figure 8: The illustration on the Data Grid environment used in this experiment. First of all, all the tasks are sent to the RB from the UI. Next, the RB allocates them to the available CEs at that time. At last, the termination of all the calculation is reported to the UI and all the results are gathered.

5.2 Working flow

The working flow of the program is shown in Figure 9.

First of all, all the sequences of the target genome is fetched from a genome database and stored on a UI machine. All the sequences are hashed and divided on the basis of their hash-key on the CE. These sequences are submitted to a RB and sent to an available CE assigned by a RB. There exist the sequences that have the same hash-key in each file. Next, every file is processed with the proposed sorting method on the CE. At this stage, the value of USL and LFO of all the sequences are calculated. These results are sent back to the UI machine and sorted based on the value of LFO.

6 Result

In this section, the calculation result on the European Data Grid is given. Some target specific sequences selected on the basis of our criterion are shown. Furthermore, the distribution of the value of USL and LFO of a gene is shown.

6.1 Unique sequences on *E. coli* genome

We tried to design the unique sequences for 4405 open reading frames (ORFs) of *E. coli* by using the proposed method. The reverse sequences of all the ORFs were also taken into consideration. Thus, the calculation was done for 8810 gene sequences. The length of the hash-key sequence was assigned to 3. The limit of the maximum length of unique sequence length was assigned to 20-mer. The sequences whose length was beyond this threshold were removed.

Table 8 shows samples of the selected sequences with our method. These sequences in itself exist only once on the target ORF sequences and its reverse ones. Namely, these sequences are unique on the whole target. These unique sequences are shown in order of LFO. As shown in this table, the order of LFO of these unique sequences does not correspond to that of their length. That is to say, a short unique

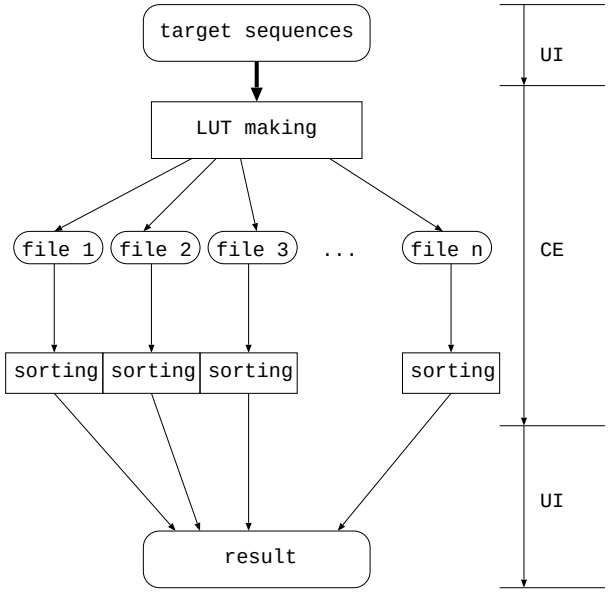

Figure 9: The illustration on the working flow of our program. At first, all the sequences of the target are fetched from a genome database as a file. Second, this file is sent on the Data Grid through the RB and all the sequences are hashed and divided into some smaller files on the basis of their hash-key on a CE. Third, these sequences are sorted on CEs. At last, all the results are reported.

sequence does not necessarily have the small value of LFO.

6.2 Distribution of LFO

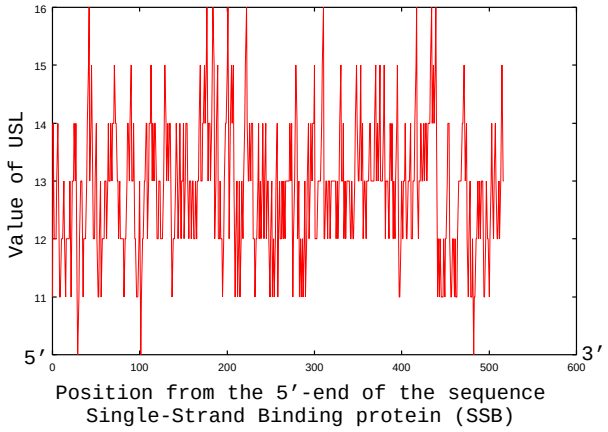

Figure 10: This is an example of the distribution of USL on the sequence of the ORF, SSB. The axis of abscissas represents the position from the 5'-end of the sequence. The axis of ordinates represents the value of USL.

Figure 10 shows an example of the distribution of USL on the sequence of an ORF. This ORF is the code region of Single-Strand Binding protein (SSB) on *E. coli* genome. The axis of abscissas represents the position from 5'-end of the sequence. The axis of ordinates represents the value of USL. The partial sequence whose value of USL is smaller is more desirable to be used as the 3'-end sequence of a primer in our previous criterion. The minimum value of USL is 10. The sequences whose USL is more than Th are omitted.

LFO	sequence
27.696700	ctgtctag
28.269300	gttcctag
28.389800	ggtgtctag
28.962400	aattcctag
28.962400	ttttcctag
29.144800	ccgtcctag
29.163000	acgtctag
29.237100	cgggtctag
29.256600	gtacctag
29.374300	agacctag
29.476900	acctctag
29.642600	gcggtctag
30.008100	ctccctag
30.036500	agactctag
30.067500	aggacctag
30.083000	gagctctag
30.170000	gccctctag
30.280000	gtatctag
30.392900	caggcctag
30.518900	acgccctag
30.518900	gtcaccctag
30.531100	atcggtcctag
30.748300	aacctagg

Table 8: An example of the selected sequences. These sequence are unique on the entire target. We extracted these sequences from the sequences selected by our method. LFO means local frequency of occurrence.

Figure 11 shows an example of the distribution of the value of LFO on the sequence of the ORF, SSB. The X axis represents the position from the 5'-end of the sequence. The Y axis represents the value of LFO calculated by the proposed method. We can observe that the order of most values of LFO conforms to that of USL. On the other hand, the value of USL of the partial sequence located around position 400 is larger than that of the partial sequence located around position 100. However, the value of LFO of the sequence located around position 400 is smaller than that of position 100. Besides that, in terms of the value of USL, partial sequences having the peak value around 200 exist. However, their values of LFO are not maximum. These mean that the unique sequences whose length is short do not always have the small value of LFO.

7 Discussion

In our formerly proposed method, the uniqueness of sequences was guaranteed on the whole target. This method deduced the minimum length of every unique sequence on the target and we used the length of the unique sequences in order to evaluate the specificity of them. In this work, we improved it to take into account the rate of occurrence of sequences. In order to realize this improvement, we introduced the value of local frequency of occurrence (LFO). Hereby, it was feasible to evaluate the unique sequences that have the same length as each other and to distinguish one from another. Moreover, this method calculates the frequency of occurrence of each sequence locally, unlike the normal frequency of occurrence method. Hence, we can say that it is possible to attach importance to the position of the components by using this method. We made much of the 3'-end region of the sequence for primer and described how to realize it.

Hereafter, we will make this method take into consideration single gap and single mismatch in the 5'-end region. Besides that, we will apply this method to probe design as well as primer design. Namely, we

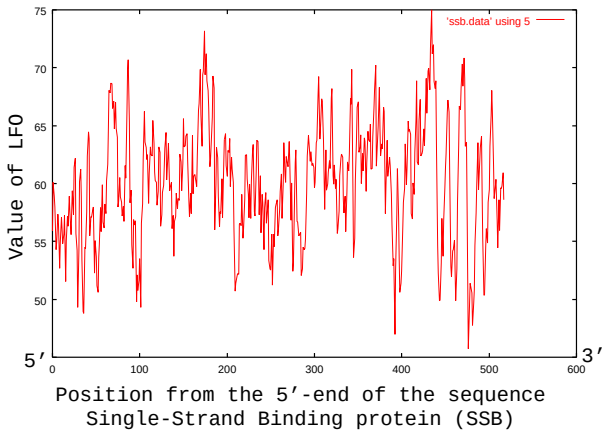

Figure 11: This is an example of the distribution of LFO on the sequence of the ORF, SSB. The axis of abscissas represents the position from the 5'-end of the sequence. The axis of ordinates represents the value of LFO.

are preparing to propose a method not only to consider the hybridization reaction in 3'-end region but also to take into account the hybridization reaction of the sequence as a whole.

As for the method to find the unique sequence, suffix tree (or suffix array) is useful and popular one. In effect, there are some methods to design optimal DNA oligo primers by using suffix array (Fugen 2001). From now on, a large amount of DNA sequences are to be unveiled more and more. The more the quantity of genomic information increases, the more the computer power is required. Hence, it will be inevitable not only to invent a rapid algorithm to analyze the information but also to develop methods to implement and process it in distributed computing environment, such as the Data Grid. In this work, we described that our method was easy to parallelize and implement in this environment. As our future work, we would like to propose a more effective method to find unique and proper sequences for primers and probes on the Data Grid. For instance, we would like to develop a method to be able to deal with and analyze all the data existing in a distributed environment at the same time. Besides that, we would like to develop the method to analyze biological important data, while implementing it on the Data Grid.

The European Data Grid infrastructure is mainly composed of UI, RB, CE and SE by default for the time being. It is said that this infrastructure is at the developing stage now. In this work, we mainly used only UI, RB and CE in order to resolve the problem. However, it seems that the quantity of stocked genomic information should double every 8 months and its rate may accelerate. As the quantity of the genomic information augments, it is being inescapable to store the information in a distributed fashion. Moreover, we must develop an effective way of dealing with such a great deal of information on the restricted network resource. Hence, we have to investigate how to process such information using SE, and how to implement the method upon the EDG.

8 Acknowledgements

We profusely appreciate to the members of the European Data Grid project, especially, the fellows of the biomedical group for their valuable advice.

We thank Yoshida Scholarship Foundation for its support during our work.

References

Allawi,H.T. & SantaLucia,J.J. (1997), 'Thermodynamics and NMR of Internal G.T Mismatches in DNA', *Biochemistry* **36**, 10581–10594.

Fugen,L. &Stormo,G.D. (2001), 'Selection of optimal DNA oligos for gene expression arrays', *Bioinformatics* **17**, 1067–1076.

Griffais,R., Andre,P.M. &Thibon,M. (1991), 'K-tuple frequency in the human genome and polymerase chain reaction', *Nucleic Acids Research* **19**, 3887–3889.

Hosaka,N., Kurata,K. &Nakamura,H. (2001), 'Comparison of Methods for Probe Design', *Genome Informatics* **12**, 449-450.

Kurata,K. &Nakamura,H. (2000), 'Novel Method for Primer/Probe Design', *Genome Informatics* **11**, 331–332.

Kurata,K., Dine,G., Saguez,C. &Nakamura,H. (2002), 'Rapid Analysis of Specificity of PCR Product on the Whole Genome', *in* 'The International Conference on Parallel and Distributed Processing Techniques and Applications', CSREA Press, Las Vegas, Nevada, USA, **1**, 246–252.

Mitsuhashi,M., Cooper,A., Ogura,M., Shinagawa,T., Yano,K. &Hosokawa,T. (1994), 'Oligonucleotide probe design - a new approach', *Nature* **367**, 759–761.

SantaLucia,J.J. (1998), 'A unified view of polymer, dumbbell, and oligonucleotide DNA nearest-neighbor thermodynamics', *Proc. Natl. Acad. Sci. U S A* **17**, 1460–1465.

Breton,V., Montagnat,J. &Medina,R. (2001), 'DataGrid, prototype of a biomedical Grid', Proceedings of the conference "Synergy between bioinformatics, medical informatics and neuroinformatics", Brussels, December 2001.

Foster,I., Kesselman,C., &Tuecke,S. (2001), 'The Anatomy of the Grid', *International Journal of High Parformance Computing Applications* **15**(3), 200–222.

Foster,I. (2002), 'The Grid: A New Infrastructure for 21st Century Science', *Physics Today* **54**(2).

Segal,B. (2000), 'Grid computing: the European data project', *IEEE Nuclear Science Symposium and Medical Imaging Conference*, Lyon, France, 15–20.

Iven,J. 'WP4 Interim Installation Solution', http://datagrid.in2p3.fr/distribution/datagrid/wp4/installation/doc/

The DataGrid Project, http://eu-datagrid.web.cern.ch/eu-datagrid/