IMPERFECTNESS OF DATA FOR STS-BASED PHYSICAL MAPPING

Hiro Ito,[1] Kazuo Iwama,[1] and Takeyuki Tamura[1]

[1]*School of Informatics, Kyoto University*
Kyoto 606-8501, Japan
{itohiro, iwama, tamura}@kuis.kyoto-u.ac.jp

Abstract In the STS-based mapping, we are requested to obtain the correct order of probes in a DNA sequence from a given set of fragments or equivalently a hybridization matrix A. It is well-known that the problem is formulated as the combinatorial problem of obtaining a permutation of A's columns so that the resulting matrix has the consecutive-one property. If the data (the hybridization matrix) is error free and includes enough information, then the above column order determines the correct order of the probes uniquely. Unfortunately this is no longer true if the data include errors, which has been one of the popular research targets in computational biology. Even if there is no error, ambiguities in the probe order may still remain. This in fact happens by the lack of some information of the data, but almost no further investigation was made previously. In this paper, we define a measure of such imperfectness of the data as a minimum amount of additional fragments which are needed to fix the probe order uniquely. Several polynomial-time algorithms to compute such additional fragments of minimum cost are presented.

Keywords: DNA, hybridization, probe, fragment and PQ-tree

1. Introduction

The STS-based mapping is one of the most popular techniques for physical mapping of DNA sequences. In this procedure, a DNA sequence S is cloned into many copies and then they are cut into smaller, overlapped subsequences called *fragments*. An STS (sequence-tagged site), also called a *probe*, is used as a marker; each probe is supposed to appear at a unique position in the entire DNA sequence S. Now we are given a *hybridization matrix*, an *H-matrix* in short, $A = (a_{ij})$ such that $a_{ij} = 1$ if probe p_j exists in fragment f_i and $a_{ij} = 0$ otherwise. Our goal is to compute the order of probes P=$\{p_1, \ldots, p_n\}$ in the original DNA sequence S from the given H-matrix A. It is well-known that this can be formulated as the following combinatorial problem: Given an H-

matrix, obtain a permutation of the columns so that the resulting matrix has the so-called *consecutive-one property*, i.e., all 1s are consecutive in each row of the matrix.

The problem can be solved in linear time by using the famous data stracture called *PQ-trees* [12]. Unfortunately, there are several kinds of errors involved in experiments, which makes the data, H-matrices in our case, imperfect. Typical errors include the case that (i) an entry of the H-matrix changes from 0 to 1, and vice versa, and that (ii) two fragments which are not consecutive in the DNA sequence put together into a "chimeric" fragment [3, 4, 6–9]. In the presence of such noises, we cannot use PQ-trees any longer; the problem now becomes several optimization problems due to different assumptions of noises. Not surprizingly, they are NP-hard in most cases [3, 7, 9, 11].

Even if there are no such errors, there may still remain ambiguities in the probe order. See for example Fig. 1 (a), which illustrates an example of an H-matrix consisting of six fragments (rows) f_1 to f_6, and ten probes (columns) A to J. By exchanging columns, the matrix can be transformed into the matrix in Fig. 1 (b) which satisfies the consecutive-one property, (i.e., each row has a single block of consecutive ones). One can see however that there are several other orders of the columns, say EGBFIADHC, which also achieve the consecutive-one property. Thus we cannot fix the order of probes uniquely from the requirement of the consecutive-one property in the case of this H-matrix, which is obviously due to the imperfectness of the data. There is a few literature which mentions the existence of this fact, e.g., [2], but no further investigation was made previously.

Our contribution. In this paper, we propose a measure of such imperfectness in H-matrices. Recall that the imperfectness is due to the lack of information. For example, if we add two extra fragments to the H-matrix of Fig. 1 (a) as in Fig. 1 (c), then the order of probes is now determined uniquely as shown in Fig. 1 (d). Thus the amount of additional fragments being needed to uniquely fix the probe order looks closely related to the degree of the imperfectness. It is apparently convenient to know this quantity for conducting the STS-based physical mapping.

More formally we consider the following problem: For a given H-matrix, obtain the minimum amount of additional fragments such that there is only one order of columns for the augmented H-matrix to have the consecutive-one property. Here are some issues which should be taken into consideration: (i) The minimum amount of fragments differs according to the order of probes to be selected as a unique one among possible different orders. For example, we needed two additional fragments in Fig. 1 (d), but three additional fragments are needed to fix the column order as BGEAIDHCFJ. (ii) There are different measures for the "amount" of fragments, such as the number of fragments and the total length of them.

Our main result is to give polynomial-time algorithms which compute (1) for a given H-matrix having the consecutive-one property, the minimum number of additional fragments which are enough to fix the probe order to the current order (i.e., the order of the columns in the given H-matrix), (2) the minimum total length of additional fragments under the same setting, (3) for a given H-matrix not necessary having the consecutive-one property, the minimum number of additional fragments enough to fix the probe order uniquely (but the order itself may be arbitrary) so that the augmented H-matrix has the consecutive-one property, (4) the minimum total length of additional fragments under the same setting. We also mention a computer simulation using genes of human chromosome 20.

Related Work. As mentioned previously, if the data are perfect, then the problem can be solved in linear time by using PQ-trees [12]. Several possibilities of errors have been investigated including obtaining a sub-matrix have the consecutive-one property [1], obtaining most-likely probe orders in the presence of false position and false negative hybridization errors using a different data structure [7], using the LP-relaxation for optimizing the most-likely probe order [6], and exploiting the fact that each probe occurs at a unique position in a more sophisticated way to handle errors such as chimeric fragments [3]. Also see [4, 8–11] for other related work including parallelization of the construction of PQ-trees [5].

2. PQ-trees

PQ-trees are a convenient data structure for our problem. Fig. 3 shows an example of a PQ-tree. A PQ-tree T consists of *P-nodes* denoted by circles, *Q-nodes* denoted by rectangles, and *leaf-nodes*. $P(T)$ denotes a set of permutations of leaf-nodes that is defined by the following rules: (i) Children of a P-node may be arbitrarily permuted. (ii) Children of a Q-node must be consecutive but may be arranged in reverse order. For example, let T_0 be the PQ-tree in Fig. 3. Then $P(T_0)$=$\{BGEJAIDHCJF, EGBJIADHCF, \ldots\}$. Two PQ-trees T and T' are said to be equivalent if $P(T) = P(T')$.

There is a linear-time algorithm [12] which constructs a PQ-tree T from H-matrix A such that (i) T's leaf-nodes correspond to columns of A and (ii) A has the consecutive-one property iff A's columns are rearranged into an order in $P(T)$. (If A cannot be rearranged into any matrix having the consecutive-one property, then the algorithm can detect it. If A is an H-matrix, this does not happen unless A includes errors. Although details are omitted, the algorithm constructs a target PQ-tree by transforming PQ-trees step-by-step, begining with a PQ-tree of a single P-node. In each step, a row of the H-matrix is selected and the PQ-tree changes so that the constraint by that row is added. For example, from the H-matrix in Fig 1(a), we can construct the associated

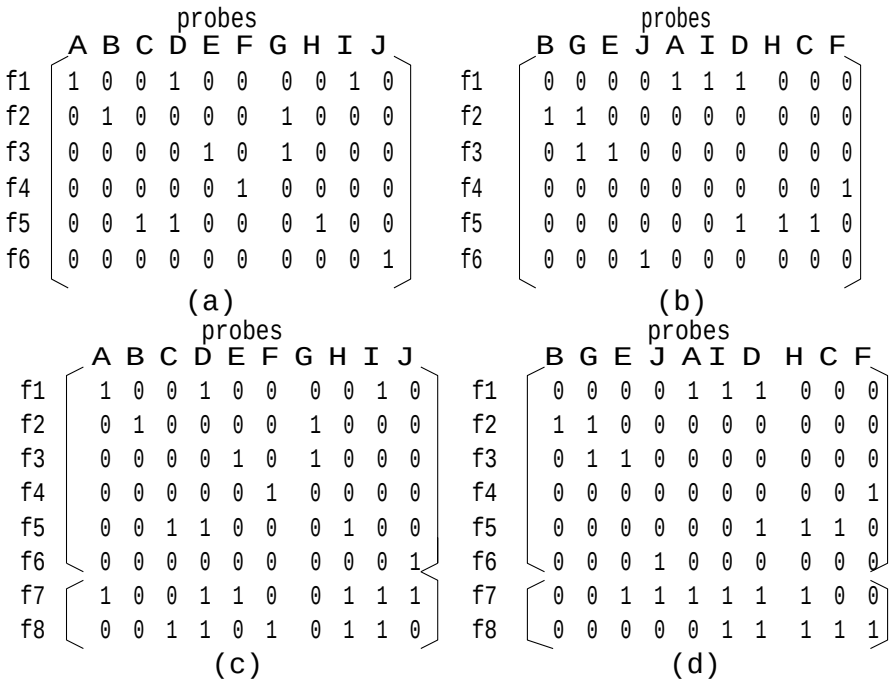

Figure 1. Permuting the (0,1)-matrix gotten by the experiment

PQ-tree as shown in Fig. 2 by selecting rows f_1 through f_6 in each step. Note that the final PQ-tree is the same as T_0 in Fig. 3 and $P(T_0)$ includes several different orders as mentioned before. For example, BGEJAIDHCF in $P(T_0)$ corresponds to the H-matrix in Fig. 2 (b) which has the consecutive-one property.

If we add two new rows (fragments) f_7 and f_8 as in Fig. 2 (c), then the PQ-tree is furthermore changed as in Fig. 4 and the final PQ-tree consists of a single Q-node (Such a PQ-tree is called a *1Q-tree*.) This means that the probe order is fixed uniquely (without its reverse order) by adding two extra fragments, which is exactly what we wanted to do. Thus our problem can be restated as follows.

Problem $FIX(T, \sigma)$: For a given PQ-tree T (made from H-matrix by the algorithm of [12]) and a probe order (leaf order) σ, obtain a set of additional fragments of a minimum cost such that T will change into a 1Q-tree of leaf order σ.

If σ is not given then the problem is denoted by $FIX(T, -)$ which requires to obtain a minimum fragments to change T into *some* 1Q-tree. As the cost of a fragment set, we consider mainly two different definitions. One is the size of fragment set, i.e., the number of fragments. The other is the sum of the lengths

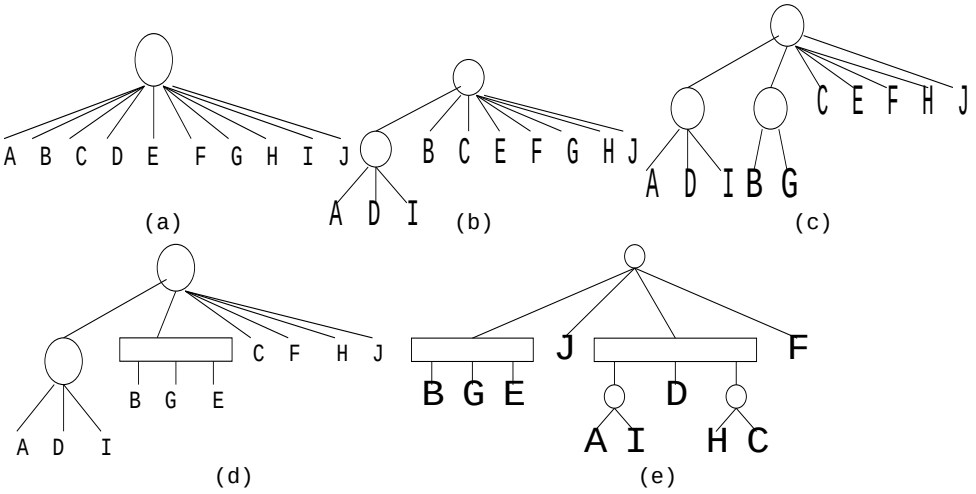

Figure 2. The process to make the input PQ-tree

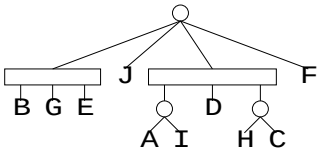

Figure 3. After adding fragment 5

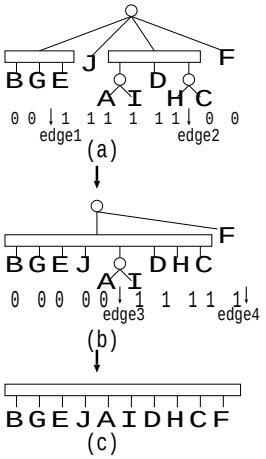

Figure 4. Making a 1Q-tree

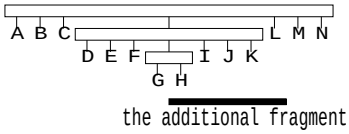

Figure 5. The number of fragment=1
the total length=5

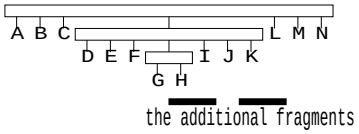

Figure 6. The number of fragment=2
the total length=4

of the fragments, where the length of a fragment is the number of 1s included in the corresponding row of the H-matrix. Those two measures sometimes conflict: As a simple example, the PQ-tree in Fig. 5 needs one fragment of length five (i.e., the fragment including probes H, I, J, K and L) for the fixing operation. The same fixing operation is also possible by using two fragments of total length four as shown in Fig. 6.

3. Minimizing the number of additional fragments

In this section, we first discuss minimizing the number of additional fragments for $FIX(T, \sigma)$ (i.e., the probe order is to be fixed to σ which is explicitly given) and then for $FIX(T, -)$ (to be fixed to an arbitrary order).

$\boldsymbol{FIX(T, \sigma)}$

Suppose that the PQ-tree is given so that the leaves are arranged in the order $\sigma = p_1 p_2 \ldots p_n$ of length n. Then we consider $n + 1$ different *positions*, denoted by $(-, p_1), (p_1, p_2), \ldots, (p_{n-1}, p_n)$, and $(p_n, -)$. Thus a position means a "between" of two consecutive probes or the left (right) of p_1 (p_n). A position denoted by (p_i, p_{i+1}) is called an *inside* position, $(-, p_1)$ and $(p_n, -)$ an *outside* position. See Fig. 4 again. An additional fragment should have a consecutive sequence of probes, EJAIDH for example for the first added fragment in Fig. 4, which can be designated by giving two positions, its *left end-position* and *right end-position* ((G,E) and (H,C)) in the example). We sometimes say that a fragment is *terminated* by its (left and/or right) end-positions.

In Fig. 4, we selected two positions (G,E) and (H,C) to terminate the first additional fragment. As one can see later, this selection of (G,E) and (H,C) contributes to converting the PQ-tree into the final 1Q-tree efficiently. Thus among all positions, there are some "important" positions for our purpose. We call such positions "edges," since using these important positions as edges of additional fragments plays a great role in minimizing the number of additional fragments. *Edges* are divided into three types and defined as follows: A position (x, y) is called (i) an *Inside-P-type edge* if probes x and y are children of a single P-node, (ii) an *Outside-P-type edge* if probe x (or y) is $-$ and it is a child of the root P-node, (iii) a *Q-type edge* if both x and y belong to a single Q-node which is not a root Q-node and which includes only leaf-nodes. In Fig. 4 for example, (A,I) is Inside-P-type, (F,-) is Outside-P-type and (G,E) is Q-type. It should be noted that if we select two edges appropriately to terminate an additional fragment, like (G,E) and (H,C) in Fig. 4 then those two edges "disappear" in the transformed PQ-tree. ((G,E) or any other Q-type edge for the Q-node BGE. By definition, (B,G) is also a Q-type edge for the same Q-node. As described later, we only need one Q-type edge for a Q-node for the

fixing operation.) Thus the key point is how to select such appropriate edges for additional fragments.

LEMMA 1 *A PQ-tree includes no edge if and only if it is a 1Q-tree.*

Proof. If a PQ-tree has two internal nodes, there is at least one edge by the definition. If a PQ-tree has only one internal node and if it is a P-node, it includes at least one P-type edge from the definition. □

LEMMA 2 *For any solution of $FIX(T, \sigma)$, every edge must be selected at least once to terminate additional fragments.*

Proof. It is proved by examing all templates for transformation of PQ-trees in each step defined in [12]. □

In Fig. 4, the first additional fragment is terminated by edges (G,E) and (H,C). After adding this fragment, edges 1 and 2 disappear. However, we cannot say that every edge always disappears when a fragment terminated by the edge is added. For example, if the first additional fragment is terminated by (A,I) and (H,C), two Inside-P-type edges seem to disappear. However, because (A,I), (I,D), (D,H) and (H,C) become Q-type edges, the number of edges which are disappeared by this additional fragment is actually only one. In Fig. 4, edges 1 and 2 have another edge, edge 3, between them. In fact, both edges always disappear in such a case as shown in the following lemma.

LEMMA 3 *Suppose that a PQ-tree T_1 has two edges e_1 and e_2, and T_1 is transformed into T_2 by adding the fragment terminated by e_1 and e_2. Then (i) at least one of e_1 and e_2 disappears in T_2 and (ii) if there is another edge, say e_3, between e_1 and e_2, then both e_1 and e_2 disappear in T_2 (iii) Furthermore no new edges are created.*

Proof. Let v be the lowest common ancestor of e_1 and e_2. Let v_l be the internal node which is an ancestor of e_1 and a child of v. Let v_r be the internal node which is an ancestor of e_2 and a child of v. Let l_1 be the leftest probe included in the subtree whose root is v_l. Let l_r be the rightest probe included in the subtree whose root is v_r. (See Fig. 7)

Assume that a fragment terminated by e_1 and e_2 is added and at most one of the two edges disappear. In this case, there are the following two cases only.

- When v is a P-node and there is not another edge except for e_1 and e_2 in the position set between l_l and l_r,
- When v is a Q-node and there is not another edge except for e_1 and e_2 in the positions included by the subtree whose root is v.

Our method avoids these cases. Hence (i) and (ii) are shown. Property (iii) can be proved by examining all templates for transformation of PQ-trees in each step defined in [12]. □

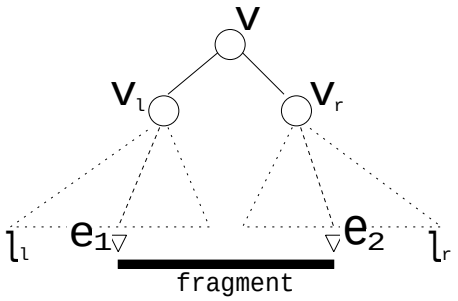

Figure 7. An additional fragment terminated by e_1 and e_2

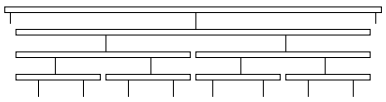

Figure 8. A PQ-tree in which $\frac{e}{2}+1$ additional fragments are necessary

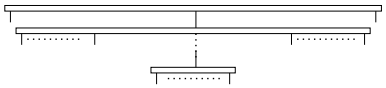

Figure 9. One step before the 1Q-tree

By using Lemma 3, we can remove two edges by adding one fragment, and thus we can show that the number of necessary additional fragments for a fixing operation is about a half of the number of edges. Note that, there must be at least three edges in order to apply lemma 3. In fact, there exists a case that there are only two edges, and two fragments are needed. As the result of this, there exists a PQ-tree that has e edges and $\lceil \frac{e}{2} \rceil + 1$ additional fragments are needed. The PQ-tree in Fig. 8 is an example of it. It becomes the PQ-tree of Fig. 9 after adding $\lceil \frac{e}{2} \rceil$ fragments terminated by two.

In other words, when the number of edges is even, there are two cases, i.e., the minimum numbers of additional fragments are $\frac{e}{2}$ and $\frac{e}{2}+1$. We can distinguish them by using a simple characterization as the following theorem.

THEOREM 4 *Let e be the number of edges and n be the number of probes of (T, σ). The minimum number of additional fragments for $FIX(T, \sigma)$ is shown as follows:*

1 *When e is odd: $\frac{e+1}{2}$.*

2 *When e is even:*

2-1. When the root node is a Q-node and there is only one internal child node of the root: $\frac{e}{2}+1$.

2-2. Otherwise: $\frac{e}{2}$.

Moreover, a fragment set with the minimum number of additional fragments for $FIX(T, \sigma)$ can be found in $O(n^3)$ time.

For proving the theorem, we introduce the following lemma.

LEMMA 5 *Consider a PQ-tree (T, σ) that includes at least three edges and doesn't satisfy the condition of 2-1 in Theorem 1. There exists a fragment satisfying the condition of Lemma 3 (ii) such that the resultant PQ-tree (T', σ) also doesn't satisfy the condition of 2-1 in Theorem 1 after adding the fragment.*

Proof of Theorem 1.

- When e is odd:

  From Lemma 3 (ii), two edges can be decreased by adding one fragment if $e \geq 3$. Hence, by iterating this process, only one edge remains after adding $\frac{e-1}{2}$ fragments. A PQ-tree including only one edge must satisfy all of the following three conditions:

  - The root is a Q-node.
  - Every internal node has at most one internal child node.
  - The lowest internal node (the internal node which doesn't have an internal child node.) is a Q-node.

  It becomes a 1Q-tree by adding a fragment.

- When e is even:

  By using the same discussion with the odd case above, a PQ-tree including only two edges can be obtained by adding $\frac{e}{2} - 1$ fragments. From Lemma 4, if the original PQ-tree doesn't satisfy the condition of 2-1, then the resultant PQ-tree doesn't also. Hence, it is enough to consider the case that $e = 2$. It can be easily proved by examing all cases.

Because we consider only the given order of probes, there are $O(n^2)$ fragments. A transformation by each additional fragment can be done in $O(n)$ time. □

$FIX(T, -)$

The result of Theorem 1 can be used to solve $FIX(T, -)$ also. That is, $FIX(T, -)$ can be solved by finding a leaf order σ in which the number of edges is minimum. The following Lemma 6 shows how to find such σ. In the lemma, v and l mean the number of internal child nodes and the number of child probes, respectively, of the noticed P-node.

LEMMA 6 *Let a_1 be the number of Q-nodes which don't have internal child nodes. Let a_2 be the total number of $max\{|l| - |v| - 1, 0\}$ for all P-nodes which are not the root. Let a_3 be $max\{|l| - |v| + 1, 0\}$ if the root is a P-node, or 0 otherwise. The minimum number of edges for $FIX(T, -)$ is $a_1 + a_2 + a_3$.*

It can be proved by definitions of edges. See an example of Fig. 3. If probe F is moved to the space between BGE and AIDHC, the number of edges decreases by one and it is the PQ-tree yielding the minimum solution three.

THEOREM 7 *In $FIX(T,-)$, a fragment set, in which the number of additional fragments is minimum, can be found in $O(n^3)$ time, where n is the number of probes.*

Proof. It is clear from Lemma 3 and 6, and Theorem 1. □

4. Minimizing the total length of additional fragments

In this section, we pay attention another cost function, i.e., minimizing the total length of additional fragments.

$FIX(T,\sigma)$

As shown in Fig.s 5 and 6, the smallness for the number of additional fragments and the shortness for the total length of additional fragments may conflict each other. For a fixing operation, for every edge, there must be at least one fragment terminated by the edges. However, there is a case that we can shorten the total length by using a fragment which is terminated by two non-edge positions. The fragment which consists of KL shown in Fig. 6 is an example of this. As shown in this example, "edge" is a concept related to the number of fragments, and there is scarcely any relation between edges and the total lengths of fragments.

We propose an algorithm, which scans from the leaves to the root and base on a dynamic programming, for this problem. We explain the basic ideas by using simple examples. Before the explanation, we introduce some notations as follows. A fragment *covers* position (i,j) if the fragment includes both i and j. A set F of fragments *covers* a set P of consecutive probes if for each neighbor probes $i,j \in P$, F has a fragment that covers position (i,j). If a set F of fragments doesn't cover a set P of consecutive probes, then there is at least one "cut" defined as follows. A *cut* of F for P is a position (i,j) such that $i,j \in F$ and the position is not covered by any fragment in F.

We consider a PQ-tree shown in Fig. 10 (c). It consists of only one P-node and leaves. The lengths of fragments are $2,3,3,3,\ldots,3,3,3,2$. The numbers of 1s assigned to each probes are $1,2,1,2,\ldots,2,1,2,1$. Note that the set of fragments covers the set of all probes. If additional fragment set doesn't cover the set of all probes as Fig. 10 (d), the fixing operation can't be completed. However, if this is a subtree of the given PQ-tree, although the additional fragment set doesn't cover the set of all probes, there is a case that the fixing operation can be completed. In many cases, it causes to save the total length of additional fragments. In other words, a naive procedure such as to

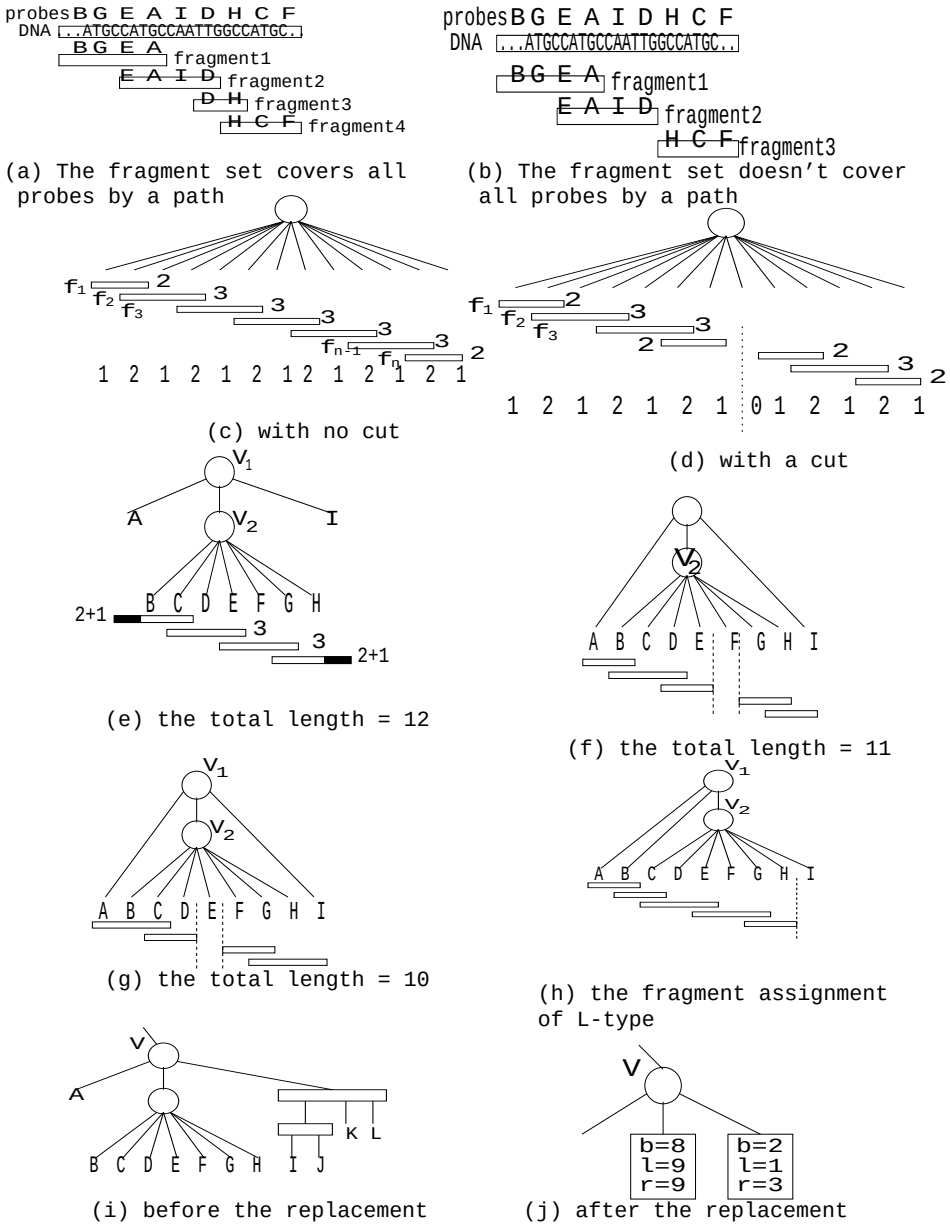

Figure 10. minimizing the total length of additional fragments

find the optimal solution in each subtree and to build them up from leaves to the root simply may not to obtain an optimal solution.

For example, if the cut is moved to F as Fig. 10 (f), the total length increases from 10 to 11.

For example, the additional fragment set shown in Fig. 10 (g) doesn't cover all probes by a path in the subtree whose root is v_2. However, the subtree is also fixed by assigning fragments to A and I which are next to the subtree as in the figure. Here, we pay attention only to the subtree v_2 (the subtree rooted by v_2) of Figures 10 (e)–(g). If additional fragments on the subtree are given as Fig. 10 (g), order of BCDE, E, and FGH cannot be fixed yet. Thus fragments A and I, which are neighbors of the subtree, must be covered by fragments.

Hence, let us say that the pair of such subtree and such fragment assignment is B-type (B means "both sides".The precise definition will be done later). Moreover, if the fragment assignment on the subtree v_2 are given as Fig. 10 (h), we have to assign a fragment to B which is the left neighbor of the subtree. Hence, let us say that the pair of such subtree and such fragment assignment is L-type. R-type is defined symmetrically. More precisely, they are defined as follows: (Note that a pair of a subtree and a fragment assignment can be two or three types at a time.)

- **R-type** A pair of a subtree and a fragment assignment, such that if there is 1 at the right neighbor probe of the subtree, the subtree can be transformed into 1Q-tree and connected to the right side.
- **L-type** A pair of a subtree and a fragment assignment, such that if there is 1 at the left neighbor probe of the subtree, the subtree can be transformed into 1Q-tree and connected to the left side.
- **B-type** A pair of a subtree and a fragment assignment, such that if there are 1s at the both neighbor probes of the subtree, the subtree can be transformed into 1Q-tree and connected to the both sides.

The minimum value of the total length of feasible fragment assignments for each of the three types can be calculated in polynomial time, since if the cut is fixed, then the minimum value can be obtained easily. By memorizing the minimum values of the total length of additional fragments for each of the three types for every subtree, we can also calculate the minimum values of them for the upper subtrees. Now, we establish an algorithm, which examines all candidates of the cut and finds the optimal fragment assignments in the three types for every sub-tree, in order to find the minimum fragment set of the whole PQ-tree.

The following example explains the algorithm more in detail. Fig. 10 (i) can be replaced with Fig. 10 (j) by calculating the optimal fragment assignments for the three types for every subtree except for v. Let b, l and r be the minimum values for the total lengths of the additional fragment sets of B-type, L-type and R-type, respectively.

By using the algorithm, we obtain the following theorem.

THEOREM 8 *A fragment set with minimum total length for $FIX(T, \sigma)$ can be found in $O(n^2)$ time, where n is the number of probes.*

Proof. We omit the proof for that the algorithm can construct the minimum fragment set correctly. The proof for the computational time is as follows. Let d_i be the degrees for each internal nodes x_i. Since, the computational time for each internal node is at most $O({d_i}^2)$, the whole computational time is at most $\sum_{x_i} O({d_i}^2) = O(n^2)$. □

FIX(T,-)

In $FIX(T, -)$, since there is no distinction between L-type and R-type, they are called *LR-type*. Let lr be the smaller one of l and r. Although, in $FIX(T, \sigma)$ a cut is scanned from left to right, in $FIX(T, -)$ a cut is fixed. However, the algorithm has to examine all candidates of nodes for both adjacent sides of the cut and the leftest node and the rightest node of the subtree. For the other nodes, the B-type assignment in which the total length of fragments is less than any other B-type assignment is used. Since the position of the cut and whether there is a cut or not are assumed in advance, the algorithm is not allowed to make a new cut by assigning fragments to nodes. However, if the B-type assignment is replaced by another assignment which is not B-type, a new cut is created.

Leaves and internal nodes should be ordered alternately as far as possible. Although the algorithm has to examine more cases, the order of the computation time doesn't become large.

THEOREM 9 *A fragment set with minimum total length for $FIX(T, -)$ can be found in $O(n^5)$ time, where n is the number of probes.*

5. Concluding Remarks

For the problem for fixing the probe order of a given PQ-tree, we showed two polynomial time algorithms. One of them minimizes the number of additional fragments. The other minimizes the total length of additional fragments. We solved not only the problems to fix probes as a given order, but also the problems to find the best order of the probes. For treating the former cost function, we introduced an idea of "edges". We showed the minimum number of additional fragments are $\lceil \frac{e}{2} \rceil$ or $\lceil \frac{e}{2} \rceil + 1$, where e is the number of edges.

For practical use, it may be difficult to make additional fragments which we want. However, if fragments are concentrated to the part where edges exist densely, the probability that fragments which our algorithm wants are generated becomes high. In other words, the probability that edges disappear

becomes high and fixing operations are accelerated. Some results of computer experiments for this method are appeared on our web page addressed: http://www.lab2.kuis.kyoto-u.ac.jp/~tamura/tcs2004.html.

Acknowledgments

We appreciate Mr. D. Tsuchida and Mr. H. Kasahara for their cooperation.

References

[1] M. T. Hajiaghayi and Y. Ganjali, A note on consecutive ones submatrix problem, Information processing letters 83, pp. 163–166, 2002

[2] S. Heber, J. Hoheisel, and M. Vingron, Application of bootstrap techniques to physical mapping, Genomics 69, pp235–241, 2000

[3] F. Alizadeh, R. M. Karp, D. K. Weisser, and G. Zweig, Physical mapping of chromosomes using unique probes, Symposium on Discrete Algorithms, pp. 489–500, 1994

[4] J. Krececioglu, S. Shete, and J. Arnold, Reconstructing distances in physical maps of chromosomes with nonoverlapping probes, Proceedings of the fourth annual international conference on Computational molecular biology, pp. 183-192, 2000

[5] F. S. Annexstein and R. P. Swaminathan, On testing consecutive-ones property in parallel, Proceedings of the seventh annual ACM symposium on Parallel algorithms and architectures, pp. 234-243, 1995

[6] M. Jain and E. W. Myers, Algorithms for computing and integrating physical maps using unique probes, Proceedings of the first annual international conference on Computational molecular biology, pp. 151-161, 1997

[7] T. Christof and J. Kececioglu, Computing physical maps of chromosomes with nonoverlapping probes by branch-and-cut, Proceedings of the third annual international conference on Computational molecular biology, pp. 115-123, 1999

[8] R. Beigel, N. Alon , S. Kasif , M. S. Apaydin, and L. Fortnow, An optimal procedure for gap closing in whole genome shotgun sequencing, Proceedings of the fifth annual international conference on Computational biology, pp. 22-30 2001

[9] T. Christof, M. Jnger, J. Kececioglu, P. Mutzel, and G. Reinelt, A branch-and-cut approach to physical mapping with end-probes, Proceedings of the first annual international conference on Computational molecular biology, pp. 84-92, 1997

[10] D. B. Wilson, D. S. Greenberg, and C. A. Phillips, Beyond islands (extended abstract): Runs in clone-probe matrices, Proceedings of the first annual international conference on Computational molecular biology, pp. 320-329, 1997

[11] A. Ben-Dor and B. Chor, On constructing radiation hybrid maps (extended abstract), Proceedings of the first annual international conference on Computational molecular biology, pp. 17-26, 1997

[12] K. S. Booth and G. S. Lueker, Testing for the Consecutive Ones Property, Interval Graphs, and Graph Planarity Using PQ-Tree Algorithms, Journal of Computer and System Sciences 13, pp. 335–379, 1976.

[13] A.V.Aho, J.E.Hopcraft, and J.D.Ullman, The Design and Analysis of Computer Algorithms, Addison-Wesley, Reading, Mass., 1974.