数理解析研究所講究録 1375 巻 2004 年 28-34

Canonical Data Structure for Probe Interval Graphs

R. Uehara (上原 隆平)
Natural Science Faculty, Komazawa University (駒澤大学 自然科学教室).
uehara@komazawa-u.ac.jp

Abstract

Probe interval graphs are introduced to deal with the physical mapping and sequencing of DNA as a generalization of interval graphs. Polynomial time recognition algorithms for the graph class are known. However, the complexity of the graph isomorphism problem for the class is still unknown. In this paper, extended $\mathcal{MPQ}$-trees are proposed to represent the probe interval graphs. The extended $\mathcal{MPQ}$-tree for given probe interval graph can be constructed in $O(n^2+nm)$ time. An extended $\mathcal{MPQ}$-tree is canonical, and hence we can solve the graph isomorphism problem for the graphs in $O(n^2+nm)$ time. Using the tree, we can determine that any two nonprobes are independent, overlapping, or their relation cannot be determined without an experiment. Therefore, we can heuristically find the best nonprobe that would be probed in the next experiment. Also, we can enumerate all possible affirmative interval graphs for given probe interval graph.

Keywords: Bioinformatics, graph isomorphism, probe interval graph.

1 Introduction

The class of interval graphs was introduced in the 1950's by Hajös and Benzer independently. Since then a number of interesting applications for interval graphs have been found including to model the topological structure of the DNA molecule, scheduling, and others (see [8, 15, 5] for further details). The class of probe interval graphs is introduced by Zhang in the assembly of contigs in physical mapping of DNA, which is a problem arising in the sequencing of DNA (see [17, 19, 18, 15] for background). A probe interval graph is obtained from an interval graph by designating a subset P of vertices as *probes*, and removing the edges between pairs of vertices in the remaining set N of *nonprobes*. That is, on the model, only partial overlap information is given. A few efficient algorithms for the class are known; the recognition algorithms [11, 14, 10], and an algorithm for finding a tree 7-spanner (see [4] for details). The recognition algorithm in [11] also gives a data structure that represents all possible permutations of the intervals of a probe interval graph.

A data structure called $\mathcal{PQ}$-trees was developed by Booth and Lueker to represent all possible permutations of the intervals of an interval graph [3]. Korte and Möhring simplified the algorithm by introducing $\mathcal{MPQ}$-trees [12]. An $\mathcal{MPQ}$-tree is canonical; that is, given two interval graphs are isomorphic if and only if their corresponding $\mathcal{MPQ}$-trees are isomorphic. However, there are no canonical $\mathcal{MPQ}$-trees for probe interval graphs. Given probe interval graph, there are several non-isomorphic interval graphs such that their interval representations are consistent to the probe interval graph.

In this paper, we extend $\mathcal{MPQ}$-trees to represent probe interval graphs. An extended $\mathcal{MPQ}$-tree is canonical, and it can be constructed in $O(n^2+nm)$ time. Thus the graph isomorphism (GI) problem for probe interval graphs can be solved in $O(n^2+nm)$ time. From the theoretical point of view, the complexity of the GI problem of probe interval graphs was not known (see [16] for related results and references). Thus the result improves the upper bound of the graph classes such that the GI problem can be solved in polynomial time.

From the practical point of view, the extended $\mathcal{MPQ}$-tree is very informative, which is beneficial in the Computational Biology community. The extended $\mathcal{MPQ}$-tree gives the information between nonprobes in linear time; the relation of two nonprobes is either (1) independent, (2) overlapping, or (3) not determined without experiments. Hence it is sufficient to experiment on the nonprobes in the case (3) to clarify the structure of the DNA sequence. Moreover, we can find the nonprobe v that has most nonprobes u such that v and u are in the case (3). Therefore, we can heuristically find the "best" nonprobe to fix the structure of the DNA sequence. The extended $\mathcal{MPQ}$-tree also represents all possible permutations of the intervals of a probe interval graph as in [11].

Due to space limitation, all proofs and some figures are omitted and can be found in a full draft available at **http://www.komazawa-u.ac.jp/~uehara/ps/MPQpig.pdf**.

2 Preliminaries

An edge which joins two vertices of a cycle but is not itself an edge of the cycle is a *chord* of that cycle. A graph is *chordal* if each cycle of length at least 4 has a chord. Given graph $G=(V,E)$, a vertex $v \in V$ is *simplicial* in G if $G[N(v)]$ is a clique in G.

Lemma 1 For any chordal graph, all simplicial vertices can be found in linear time.

Interval graph representation: A graph (V, E) with $V = \{v_1, v_2, \cdots, v_n\}$ is an *interval graph* if there is a set of intervals $\mathcal{I} = \{I_{v_1}, I_{v_2}, \cdots, I_{v_n}\}$ such that $\{v_i, v_j\} \in E$ iff $I_{v_i} \cap I_{v_j} \neq \emptyset$ for each i and j with $1 \leq i, j \leq n$. We call the set $\mathcal{I}$ of intervals *interval representation* of the graph. For each interval I, we denote by $R(I)$ and $L(I)$ the right and left endpoints of the interval, respectively (hence we have $L(I) \leq R(I)$ and $I = [L(I), R(I)]$).

A graph $G = (V, E)$ is a *probe interval graph* if V can be partitioned into subsets P and N (corresponding to the *probes* and *nonprobes*) and each $v \in V$ can be assigned to an interval I_v such that $\{u, v\} \in E$ iff both $I_u \cap I_v \neq \emptyset$ and at least one of u and v is in P. In this paper, we assume that P and N are given, and then we denote by $G = (P, N, E)$. Let $G = (P, N, E)$ be a probe interval graph. Let E^+ be a set of edges $\{t_1, t_2\}$ with $t_1, t_2 \in N$ such that there are two probes v_1 and v_2 in P such that $\{v_1, t_1\} \in E$, $\{v_1, t_2\} \in E$, $\{v_2, t_1\} \in E$, $\{v_2, t_2\} \in E$, and $\{v_1, v_2\} \notin E$. In the case, we have $I_{t_1} \cap I_{t_2} \neq \emptyset$. Each edge in E^+ is called an *enhanced edge*, and the graph $G^+ := (P, N, E \cup E^+)$ is said to be an *enhanced probe interval graph*. For further details and references can be found in [5, 15].

For given (enhanced) probe interval graph G, an interval graph G' is said to be *affirmative* iff G' gives one possible interval representation of G.

Given enhanced probe interval graph $G^+ = (P, N, E \cup E^+)$, let u and v be any two nonprobes with $\{u, v\} \notin E^+$. Then, we say that u *intersects* v if $I_u \cap I_v \neq \emptyset$ for all affirmative interval graphs of G^+. The nonprobes u and v are *independent* if $I_u \cap I_v = \emptyset$ for all affirmative interval graphs of G^+. Otherwise, we say that the nonprobe u *potentially intersects* v. Intuitively, if u potentially intersects v, we cannot determine their relation without experiments.

$\mathcal{PQ}$-trees and $\mathcal{MPQ}$-trees: $\mathcal{PQ}$-trees were introduced by Booth and Lueker [3], and which can be used to recognize interval graphs as follows. A *$\mathcal{PQ}$-tree* is a rooted tree T with two types of internal nodes: $\mathcal{P}$ and $\mathcal{Q}$, which will be represented by circles and rectangles, respectively. The leaves of T are labeled 1-1 with the maximal cliques of the interval graph G. The *frontier* of a $\mathcal{PQ}$-tree T is the permutation of the maximal cliques obtained by the ordering of the leaves of T from left to right. $\mathcal{PQ}$-tree T and T' are *equivalent* if one can be obtained from the other by applying the following rules a finite number of times; arbitrarily permute the successor nodes of a $\mathcal{P}$-node, or reverse the order of the successor nodes of a $\mathcal{Q}$-node. A graph G is an interval graph iff there is a $\mathcal{PQ}$-tree T whose frontier represents a consecutive arrangement of the maximal cliques of G. If G is an interval graph, then all consecutive arrangements of the maximal cliques of G are obtained by taking equivalent $\mathcal{PQ}$-trees.

Lueker and Booth [13], and Colbourn and Booth [6] developed labeled $\mathcal{PQ}$-trees in which each node contains information of vertices as labels. Their labeled $\mathcal{PQ}$-trees are *canonical*; given interval graphs G_1 and G_2 are isomorphic iff corresponding labeled $\mathcal{PQ}$-trees T_1 and T_2 are isomorphic.

$\mathcal{MPQ}$-trees are developed by Korte and Möhring to simplify the construction of $\mathcal{PQ}$-trees [12]. The $\mathcal{MPQ}$-tree T^* assigns sets of vertices to the nodes of a $\mathcal{PQ}$-tree T representing an interval graph $G = (V, E)$. A $\mathcal{P}$-node is assigned only one set, while a $\mathcal{Q}$-node has a set for each of its sons (ordered from left to right according to the ordering of the sons).

For a $\mathcal{P}$-node $\hat{P}$, this set consists of those vertices of G contained in all maximal cliques represented by the subtree or $\hat{P}$ in T, but in no other cliques[1]. For a $\mathcal{Q}$-node $\hat{Q}$, the definition is more involved. Let $Q_1, \cdots, Q_m$ $(m \geq 3)$ be the set of the sons (in consecutive order) of $\hat{Q}$, and let T_i be the subtree of T with root Q_i. We then assign a set S_i, called *section*, to $\hat{Q}$ for each Q_i. Section S_i contains all vertices that are contained in all maximal cliques of T_i and some other T_j, but not in any clique belonging to some other subtree of T that is not below $\hat{Q}$. The $\mathcal{MPQ}$-tree directly corresponds to the labeled $\mathcal{PQ}$-tree; the sets of vertices assigned in the $\mathcal{MPQ}$-tree directly correspond to the "characteristic nodes" in [6]. Thus the $\mathcal{MPQ}$-tree is canonical (although it does not shown explicitly in [12]). Thus the graph isomorphism problem for interval graphs can be solved in linear time using the $\mathcal{MPQ}$-trees, which can be obtained without constructing $\mathcal{PQ}$-trees in [3]. The property of $\mathcal{MPQ}$-trees for interval graphs is summarized as follows:

Theorem 2 Let T^* be the canonical $\mathcal{MPQ}$-tree for given interval graph $G = (V, E)$. (a) T^* can be obtained in $O(|V| + |E|)$ time and $O(|V|)$ space. (b) Each maximal clique of G corresponds to a path in T^* from the root to a leaf, where each vertex $v \in V$ is as close as possible to the root. (c) In T^*, each vertex v appears in either one leaf, one $\mathcal{P}$-node, or consecutive sections S_i, S_{i+1}, $\cdots$, S_{i+j} (with $j > 0$) in a $\mathcal{Q}$-node. (d) The root of T^* contains all vertices belonging to all maximal cliques, while the leaves contain the simplicial vertices.

Lemma 3 Let $\hat{Q}$ be a $\mathcal{Q}$-node in the canonical $\mathcal{MPQ}$-tree. Let $S_1, \cdots, S_k$ (in this order) be the

[1] We will use $\hat{P}$, $\hat{Q}$, and $\hat{N}$ for describing a $\mathcal{P}$-node, $\mathcal{Q}$-node, any node, respectively to distinguish probe set P and nonprobe set N.

sections of $\hat{Q}$, and let U_i denote the set of vertices occurring below S_i with $1 \leq i \leq k$. Then we have the following; (a) $S_{i-1} \cap S_i \neq \emptyset$ for $2 \leq i \leq k$, (b) $S_1 \subseteq S_2$ and $S_k \subseteq S_{k-1}$, (c) $U_1 \neq \emptyset$ and $U_k \neq \emptyset$, (d) $(S_i \cap S_{i+1}) \setminus S_1 \neq \emptyset$ and $(S_{i-1} \cap S_i) \setminus S_k \neq \emptyset$ for $2 \leq i \leq k-1$, (e) $S_{i-1} \neq S_i$ with $2 \leq i \leq k-1$, and (f) $(S_{i-1} \cup U_{i-1}) \setminus S_i \neq \emptyset$ and $(S_i \cup U_i) \setminus S_{i-1} \neq \emptyset$ for $2 \leq i \leq k$.

Extended $\mathcal{MPQ}$-trees: If given graph is an interval graph, the corresponding $\mathcal{MPQ}$-tree is uniquely determined up to isomorphism. However, for a probe interval graph, this is not in the case. For example, consider a probe interval graph $G = (P, N, E)$ with $P = \{1,2,3,4,5,6,7,8,9\}$ and $N = \{a, b, c, d, e, f, g\}$ given in Fig. 1. If the graph does not contain the nonprobe g, we have the canonical $\mathcal{MPQ}$-tree in Fig. 2. However, the graph is a probe interval graph and we do not know if g intersects b and/or c since they are nonprobes. According to the relations between g and b and/or c, we have four possible $\mathcal{MPQ}$-trees that are affirmative to G shown in Fig. 3, where X is either $\{1,2,7,8\}$, $\{1,2,7,8,c\}$, or $\{1,2,7,8,b,c\}$. We call such a vertex g *floating leaf* (later, it will be shown that such a vertex has to be a leaf in an $\mathcal{MPQ}$-tree). For a floating leaf, there is a corresponding Q-node (which also will be shown later). Thus we extend the notion of a Q-node to contain the information of the floating leaves. A floating leaf appears consecutive sections of a Q-node $\hat{Q}$ as the ordinary vertices in $\hat{Q}$. To distinguish them, we draw them over the corresponding sections; see Fig. 4. Further details will be discussed in Section 3.

3 Construction of Extended $\mathcal{MPQ}$-tree of Probe Interval Graph

Let $G = (P, N, E)$ be a given probe interval graph, and $G^+ = (P, N, E \cup E^+)$ be the corresponding enhanced probe interval graph, where E^+ is the set of enhanced edges. In our algorithm, simplicial nonprobes play an important role; we partition the set N of nonprobes to two sets N^* and N_S defined as follows; $N_S := \{u | u \text{ is simplicial in } G^+\}$, and $N^* := N \setminus N_S$. For example, for the graph $G = (P, N, E)$ in Fig. 1, $E^+ = \{\{c, d\}, \{c, f\}\}$, $N_S = \{a, e, g\}$, and $N^* = \{b, c, d, f\}$. The outline of the algorithm is as follows:

A0. Given probe interval graph $G = (P, N, E)$, compute the enhanced probe interval graph $G^+ = (P, N, E \cup E^+)$;

A1. Partition N into two subsets N^* and N_S;

A2. Construct the $\mathcal{MPQ}$-tree T^* of $G^* = (P, N^*, E^*)$, where E^* is the set of edges induced by $P \cup N^*$ from G^+;

A3. Embed each nonprobe v in N_S into T^*.

Note that the tree constructed in step A2 is an *ordinary* $\mathcal{MPQ}$-tree. In step A3, it will be modified to the extended $\mathcal{MPQ}$-tree. The following observation is obtained by definition:

Observation 4 Let v be a nonprobe in N_S. Then for any two vertices $u_1, u_2 \in N_{G^+}(v)$, $I_{u_1} \cap I_{u_2} \neq \emptyset$.

3.1 Construction of $\mathcal{MPQ}$-tree of G^*

Let $G^* = (P, N^*, E^*)$ be the enhanced probe interval graph induced by P and N^*. The following lemma plays an important role.

Lemma 5 Let u and v be any nonprobes in N^*. Then there is an interval representation of G^* such that $I_u \cap I_v \neq \emptyset$ iff $\{u, v\} \in E^+$.

The definition of (enhanced) probe interval graphs and Lemma 5 imply the main theorem in this section:

Theorem 6 The enhanced probe interval graph G^* is an interval graph.

Hereafter we call the graph $G^* = (P, N^*, E^*)$ *the backbone interval graph* of $G^+ = (P, N, E \cup E^+)$. For any given interval graph, its corresponding $\mathcal{MPQ}$-tree can be computed in linear time [12]. Thus we also have the following corollary:

Corollary 7 The $\mathcal{MPQ}$-tree T^* of G^* can be computed in linear time.

In the $\mathcal{MPQ}$-tree T^*, for each pair of nonprobes u and v, their corresponding intervals intersect iff $\{u, v\} \in E^+$. This implies the following observation.

Observation 8 The $\mathcal{MPQ}$-tree T^* gives us the possible interval representations of G^* such that two nonprobes in N^* do not intersect as possible as they can.

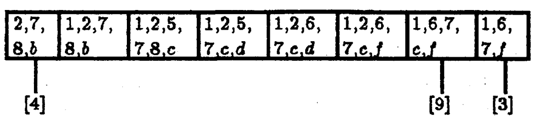

Figure 5: The canonical $\mathcal{MPQ}$-tree T^* of G^*

For example, for the graph $G = (P, N, E)$ in Fig. 1, the canonical $\mathcal{MPQ}$-tree of the backbone interval graph $G^* = (P, N^*, E^*)$ is described in Fig. 5. In the $\mathcal{MPQ}$-tree, $I_d \cap I_f = \emptyset$, while $I_d \cap I_f \neq \emptyset$ in Fig. 1.

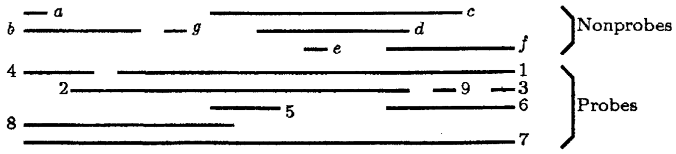

Figure 1: Given probe interval graph G

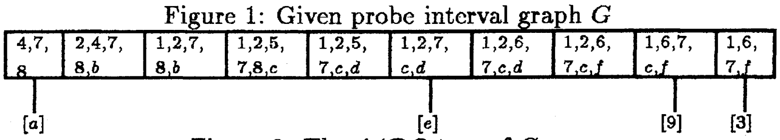

Figure 2: The $\mathcal{MPQ}$-tree of $G - g$

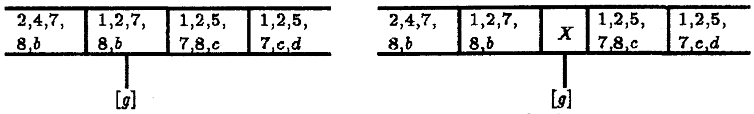

Figure 3: Four $\mathcal{MPQ}$-trees of G

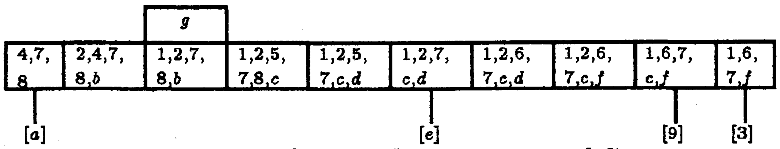

Figure 4: The extended $\mathcal{MPQ}$-tree of G

3.2 Embedding of Nonprobes in N_S

Lemma 9 For each nonprobe v in N_S, all vertices in $N(v)$ are probes.

Lemma 10 For any probe interval graph G, there is an affirmative interval graph G' such that every nonprobe v in N_S of G is also simplicial in G'.

By Lemma 10 and Theorem 2(d), we have the following corollary.

Corollary 11 For any probe interval graph G, there is an affirmative interval graph G' such that every nonprobe v in N_S of G is in a leaf of the $\mathcal{MPQ}$-tree of G'.

Our embedding is an extension of the embedding by Korte and Möhring [12] to deal with nonprobes. Each node $\hat{N}$ (including $\mathcal{Q}$-node) of the current tree T^* and each section S of a $\mathcal{Q}$-node is labeled according to how the nonprobe v in N_S is related to the probes in $\hat{N}$ or S. Nonprobes in $\hat{N}$ or S are ignored. The label is ∞, 1, or 0 if v is adjacent to all, some, or no probe from $\hat{N}$, or S, respectively. Empty sets (or the sets containing only nonprobes) obtain the label 0. Labels 1 and ∞ are called *positive* labels.

Lemma 12 For a nonprobe v in N_S, all nodes with positive labels are contained in a unique path of T^*.

Let P$'$ be the unique minimal path in T^* containing all nodes with positive label. Let P be a path from the root of the $\mathcal{MPQ}$-tree T^* to a leaf containing P$'$ (a leaf is chosen in any way). Let $\hat{N}_*$ be the lowest node in P with positive label. If P contains nonempty $\mathcal{P}$-nodes or sections above $\hat{N}_*$ with label 0 or 1, let $\hat{N}^*$ be the highest such $\mathcal{P}$-node or $\mathcal{Q}$-node containing the section. Otherwise put $\hat{N}_* = \hat{N}^*$.

When $\hat{N}_* \neq \hat{N}^*$, we have the following lemma:

Lemma 13 We assume that $\hat{N}_* \neq \hat{N}^*$. Let $\hat{Q}$ be any $\mathcal{Q}$-node with sections $S_1, \cdots, S_k$ in this order between $\hat{N}_*$ and $\hat{N}^*$. If $\hat{Q}$ is not $\hat{N}^*$, all neighbors of v in $\hat{Q}$ appear in either S_1 or S_k.

We are now ready to use the bottom-up strategy from $\hat{N}_*$ to $\hat{N}^*$ as in [12]. In our algorithm, the step A3 consists of the following substeps;

- A3.1. while there is a nonprobe v such that $\hat{N}_* \neq \hat{N}^*$ for v, embed v into T^*;
- A3.2. while there is a nonprobe v such that $\hat{N}_* = \hat{N}^*$ for v and v is not a floating leaf, embed v into T^*;
- A3.3. embed each nonprobe v (such that $\hat{N}_* = \hat{N}^*$ for v and v is a floating leaf) into T^*.

As shown later, an embedding of a nonprobe v with $\hat{N}_* \neq \hat{N}^*$ merges some nodes into one new $\mathcal{Q}$-node. Thus, during step A3.1, embedding of a nonprobe v can change the condition of other nonprobes u from "$\hat{N}_* \neq \hat{N}^*$" to "$\hat{N}_* = \hat{N}^*$". We note that A3.1 and A3.2 do not generate floating leaves, and all floating leaves are embedded in step A3.3, which will be shown later. Hence the templates used in steps A3.1 and A3.2 are not required to manage floating leaves.

Hereafter, we suppose that the algorithm picks up some nonprobe v from N_S and it is going to embed v into T^*. In most cases, the vertex set V_N of the current node or section is partitioned into A, B, and C defined as follows; $A := P \cap V_N \cap N(v)$,

$B := (P \cap V_N) \setminus A$, and $C := N \cap V_N$. Since we extend the templates in [12], we use the same names of templates as L1, P2, and so on, which is an extension of the corresponding templates in [12] (templates from Q4 to Q7 are new templates). We also use the help templates H1 and H2 in [12] if they can be applied; it is simple and omitted here. Due to space limitation, templates L1, L2, P1, P2, P3, Q1-1, Q1-2, Q3, and Q6 are omitted here. Through the embedding, we keep the following assertion:

Assertion 14 (1) Each nonprobe in N_S has no intersection with unnecessary nonprobes, (2) each leaf contains either vertices in $P \cup N^*$ or one nonprobe in N_S, and (3) each nonprobe in N_S is in a leaf.

3.2.1 Templates for nonprobe with $\hat{N}_* = \hat{N}^*$:

We first assume that $\hat{N}^* = \hat{N}_*$, which occurs in steps A3.2 and A3.3. If the node is a leaf or a $\mathcal{P}$-node, we use template L1 or P1, respectively. If $\hat{N}^* = \hat{N}_*$ is a $\mathcal{Q}$-node with sections $S_1, \cdots, S_k$ in this order, v can be a floating leaf. We let $A := (\cup_{1 \le i \le k} S_i) \cap N(v)$. Let ℓ be the minimum index with $A \subseteq S_\ell$ and r be the maximum index with $A \subseteq S_r$. That is, $A \not\subseteq S_i$ for each $i < \ell$ and $i > r$, and $A \subseteq S_j$ for each $\ell \le j \le r$. Then there are four cases:

(a) $\ell = 1$ and $A \subset S_\ell \cap P$. In the case, v may be a leaf of a new section $S_0 := A \subset S_1$. The case $r = k$ and $A \subset S_k \cap P$ is symmetric.

(b) $A = S_j \cap P$ for some $\ell \le j \le r$. In the case, v may be a leaf under the section S_j.

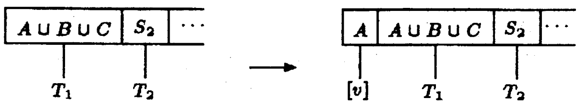

Figure 6: Template Q2 for (1) $\hat{N}_* = \hat{N}^*$ and $A \subset S_1 \cap P$, or (2) $\hat{N} = \hat{N}_* \neq \hat{N}^*$, $A \subseteq S_1$, and $A \not\subseteq \cap_{1 \le i \le k} S_i$

(c) $A = S_j \cap S_{j+1} \cap P$ for some $\ell \le j < r$. In the case, v may be a leaf under the new section $S := A \cup (S_j \cap S_{j+1} \cap N)$ between S_j and S_{j+1}.

(d) $S_j \cap S_{j+1} \cap P \subset A \subset S_j \cap P$ or $S_j \cap S_{j+1} \cap P \subset A \subset S_{j+1} \cap P$ for some $\ell \le j < r$. In the case, v may be a leaf under the new section $S := A \cup (S_j \cap S_{j+1} \cap N)$ between S_j and S_{j+1}.

The algorithm checks if the position of the v is uniquely determined. If it is uniquely determined, the algorithm embeds v into the place in step A3.2. If exactly one of the cases (a) to (d) occurs, we use a template as follows. In (a), template Q2 in Fig. 6 is used. In (b), we use one of three templates Q6-1, Q6-2, and Q6-3 as follows; if the section S_j has no child, template Q6-1 is used and v is added as a leaf under S_j; if the root of the subtree under S_j is a $\mathcal{P}$-node with empty label, template Q6-2 is used and v is added as a leaf under the $\mathcal{P}$-node; or otherwise, template Q6-3 is used and v is added as a leaf under a new $\mathcal{P}$-node with empty label under S_j. We note that Assertion 14(2) holds if $\hat{R}$ contains nonprobes. In (c) or (d), template Q2 in Fig. 6 is used (we have $A \cup (S_j \cap S_{j+1} \cap N) = S_j \cap S_{j+1}$ in (c)). We have one more case that the position of the v may be uniquely determined; $\ell = 1$, $r = k$, and $(S_i \cap S_{i+1} \cap P) \setminus A \neq \emptyset$ for each $1 \le i < k$. In the case, we use the template Q1-1. If the position is not uniquely determined, v is a floating leaf. In the case, the embedding is postponed until step A3.3, where we use template Q4 in Fig. 8 for such ℓ and r (R_i denotes the set of floating leaves in S_i). If v can be a floating leaf under some sections S (including non-existent sections), we say v can *hang down* S.

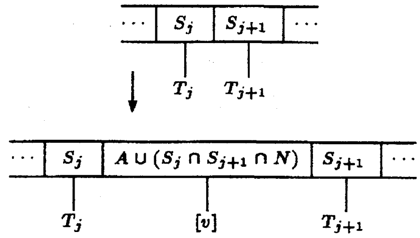

Figure 7: Template Q7 for $\hat{N}_* = \hat{N}^*$ and $S_j \cap S_{j+1} \cap P \subset A \subseteq S_{j+1} \cap P$ or $S_j \cap S_{j+1} \cap P \subset A \subseteq S_j \cap P$

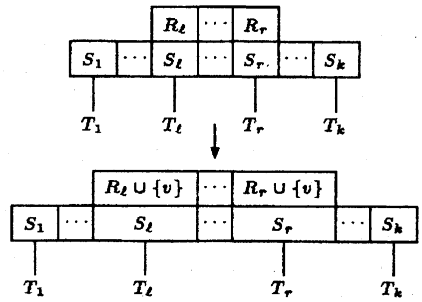

Figure 8: Template Q4 for floating leaf v

We have the following observation.

Observation 15 In steps A3.2 and A3.3, all $\mathcal{Q}$-nodes are neither divided nor merged.

3.2.2 Templates for nonprobe with $\hat{N}_* \neq \hat{N}^*$:

When $\hat{N}_* \neq \hat{N}^*$, we use the bottom-up strategy from $\hat{N}_*$ to $\hat{N}^*$ as in [12]. Let $\hat{N}$ denote the current node that starts from $\hat{N}_*$ and ends up at $\hat{N}^*$. The algorithm consists of three phases; (1) $\hat{N} = \hat{N}_*$, (2) $\hat{N} \neq \hat{N}_*$ and $\hat{N} \neq \hat{N}^*$, and (3) $\hat{N} = \hat{N}^*$. The first two phases are the extensions of the templates in [12] by Lemmas 12 and 13 which correspond to [12, Lemma 4.1]. However, the algorithm uses one more template in the third phase since Lemma 13 does not hold. The templates in the case $\hat{N}_* \neq \hat{N}^*$ never generate floating leaves. Therefore, since they are applied in step A3.1, the templates in the case are not required to manage floating leaves.

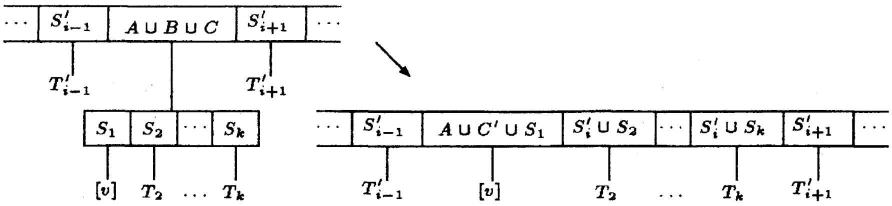

Figure 9: Template Q5 for $\hat{N} = \hat{N}^* \neq \hat{N}_*$ and $B \subseteq S'_{i+1}$

(1) $\hat{N} = \hat{N}_* \neq \hat{N}^*$. Since the label of $\hat{N} = \hat{N}_*$ is positive, $A := \hat{N} \cap N(v) \neq \emptyset$. If $\hat{N}$ is a leaf or a $\mathcal{P}$-node, the algorithm uses template L2 or P2, respectively. When $\hat{N}$ is a $\mathcal{Q}$-node, we can use Lemmas 12 and 13 in this case. Thus we have two subcases, which correspond to templates Q1 and Q2 in [12]. By Lemma 13, we assume that $A \subseteq S_1$ without loss of generality. The algorithm uses template Q1-2 if $A \subseteq S_k$, and otherwise, it uses template Q2 in Fig. 6.

Observation 16 In any case, v becomes a leaf $[v]$ under a non-empty section S_1 of a $\mathcal{Q}$-node since $A \neq \emptyset$.

(2) $\hat{N} \neq \hat{N}_*$ and $\hat{N} \neq \hat{N}^*$. If $\hat{N}$ is a $\mathcal{P}$-node, the algorithm uses template P3. If $\hat{N}$ is a $\mathcal{Q}$-node, we can use Lemmas 12 and 13 again and the algorithm uses template Q3. By a simple induction of the length of the path P with Observation 16, we again have the following observation (since $S_1 \neq \emptyset$):

Observation 17 In any case, v becomes a leaf $[v]$ under a non-empty section S_1 of a $\mathcal{Q}$-node.

(3) $\hat{N} = \hat{N}^* \neq \hat{N}_*$. If $\hat{N}$ is a $\mathcal{P}$-node, the algorithm uses the template P3 again. If $\hat{N}$ is a $\mathcal{Q}$-node, we cannot use Lemmas 13. Let S'_i be the section in $\hat{N}$ such that the subtree T'_i contains $[v]$. If S'_i is the leftmost or rightmost section in $\hat{N}$, we can use the template Q3 again. Thus we assume that $1 < i < k'$, where k' is the number of sections in the $\mathcal{Q}$-node $\hat{N}$. Let S'_{i-1} and S'_{i+1} be the left and right sections of S'_i, respectively. We now define $A := N(v) \cap S'_i$ and $B := (S'_i \cap P) \setminus A$. Then, since the label of S'_i is 0 or 1, we have $B \neq \emptyset$. For the set B, we have the following lemma:

Lemma 18 Either $B \subseteq S'_{i+1} \setminus S'_{i-1}$ or $B \subseteq S'_{i-1} \setminus S'_{i+1}$.

Without loss of generality, we assume that Lemma 18(a) occurs. That is, all vertices in B appear from the section S'_i to the some sections on the right side of S'_i. Let $C' := S'_{i-1} \cap S'_i \cap N$. That is, C' is the set of nonprobes appearing both of S'_{i-1} and S_i. Then we use template Q5 in Fig. 9. In the figure, C denotes the nonprobes in S'_i; that is, $S'_i = A \cup B \cup C$ and $C' \subseteq C$.

Example 19 For the graph $G = (P, N, E)$ in Fig. 1 with its backbone interval graph in Fig. 5, the extended $\mathcal{MPQ}$-tree $\tilde{T}$ is shown in Fig. 4. The algorithm uses templates L2 and Q3 to embed a, and uses template Q4 to embed g since it is a floating leaf. For the nonprobe e, only the case (c) in Section 3.2.1 can be applied; $\{1,2,7,8,c,d\} \cap \{1,2,6,7,c,d\} \cap P = \{1,2,7\} = N(e)$. Thus its position is uniquely determined, and embedded between the sections. Note that we can know that e intersects both of c and d with neither experiments nor enhanced edges. We also note that I_a and I_b could have intersection, but they are standardized according to Assertion 14(1).

3.3 Analysis of Algorithm

Since the correctness of steps A0, A1, and A2 follows from Theorem 6, we concentrate on step A3. First, the templates cover all formally distinct cases. All templates for the case $\hat{N}_* = \hat{N}^*$ with the help-templates H1 and H2 in [12] are easily shown to be correct. Thus we consider the case $\hat{N}_* \neq \hat{N}^*$.

Theorem 20 When $\hat{N}_* \neq \hat{N}^*$, v is not a floating leaf.

Theorem 21 The resulting extended $\mathcal{MPQ}$-tree is canonical up to isomorphism.

Theorem 22 For given probe interval graph $G = (P, N, E)$, let $\tilde{T}$ be the canonical extended $\mathcal{MPQ}$-tree, and $G^+ = (P, N, E \cup E^+)$ be the corresponding enhanced interval graph. Let $\tilde{E}$ be the set of edges $\{v_1, v_2\}$ joining nonprobes v_1 and v_2 which is given by $\tilde{T}$; more precisely, we regard $\tilde{T}$ as an ordinary $\mathcal{MPQ}$-tree, and the graph $\tilde{G} = (P \cup N, E \cup E^+ \cup \tilde{E})$ is the interval graph given by the $\mathcal{MPQ}$-tree $\tilde{T}$ (thus a floating leaf is not a leaf; the vertex appears in consecutive sections in the corresponding $\mathcal{Q}$-node). Then $\tilde{T}$ can be computed in $O((|P| + |N|)|E| + |E^+| + |\tilde{E}|)$ time and $O(|P| + |N| + |E| + |E^+| + |\tilde{E}|)$ space.

Corollary 23 The graph isomorphism problem for the class of (enhanced) probe interval graphs G is solvable in $O(n^2+nm)$ time and $O(n^2)$ space, where n and m are the number of vertices and edges of an affirmative interval graph of G, respectively.

4 Application

We consider the following problem:

Input: An enhanced probe interval graph $G^+ = (P, N, E \cup E^+)$ and the canonical extended $\mathcal{MPQ}$-tree $\tilde{T}$;

Output: Mapping f from each pair of nonprobes u, v with $\{u, v\} \notin E^+$ to "intersecting", "potentially intersecting", or "independent";

We denote by E_i and E_p the sets of the pairs of intersecting nonprobes, and the pairs of potentially intersecting nonprobes, respectively. That is, each pair of nonprobes u, v is either in E^+, E_i, E_p, or otherwise, they are independent.

Theorem 24 The sets E_i and E_p can be computed in $O(|E| + |E^+| + |E_i| + |E_p|)$ time for given enhanced probe interval graph $G^+ = (P, N, E \cup E^+)$ and the extended $\mathcal{MPQ}$-tree $\tilde{T}$.

By Theorem 24, we can heuristically find the "best" nonprobe to fix the structure of the DNA sequence:

Corollary 25 For given enhanced probe interval graph $G^+ = (P, N, E \cup E^+)$ and the canonical extended $\mathcal{MPQ}$-tree $\tilde{T}$, we can find the nonprobe v that has most potentially intersecting nonprobes in $O(|E| + |E^+| + |E_i| + |E_p|)$ time.

References

[1] C. Berge. *Hypergraphs*. Elsevier, 1989.

[2] J.R.S. Blair and B. Peyton. An Introduction to Chordal Graphs and Clique Trees. In *Graph Theory and Sparse Matrix Computation*, Vol. 56 of *IMA*, pages 1–29. Springer, 1993.

[3] K.S. Booth and G.S. Lueker. Testing for the Consecutive Ones Property, Interval Graphs, and Graph Planarity Using *PQ*-Tree Algorithms. *JCSS*, 13:335–379, 1976.

[4] A. Brandstädt, F.F. Dragan, H.-O. Le, V.B. Le, and R. Uehara. Tree Spanners for Bipartite Graphs and Probe Interval Graphs. In *WG '03*, pages 106–118. LNCS Vol. 2880, Springer-Verlag, 2003.

[5] A. Brandstädt, V.B. Le, and J.P. Spinrad. *Graph Classes: A Survey*. SIAM, 1999.

[6] C.J. Colbourn and K.S. Booth. Linear Time Automorphism Algorithms for Trees, Interval Graphs, and Planar Graphs. *SIAM J. on Comp.*, 10(1):203–225, 1981.

[7] P. Galinier, M.Habib, and C. Paul. Chordal Graphs and Their Clique Graphs. In *WG '95*, pages 358–371. LNCS Vol. 1017, Springer-Verlag, 1995.

[8] M.C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, 1980.

[9] W.-L. Hsu and T.-H. Ma. Substitution Decomposition on Chordal Graphs and Applications. In *ISA '91*, pages 52–60. LNCS Vol. 557, Springer-Verlag, 1991.

[10] J.L. Johnson, R.M. McConnell, and J.P. Spinrad. Linear Time Recognition of Probe Interval Graphs. in preparation, 2002.

[11] J.L. Johnson and J.P. Spinrad. A Polynomial Time Recognition Algorithm for Probe Interval Graphs. In *SODA '01*, pages 477–486. ACM, 2001.

[12] N. Korte and R.H. Möhring. An Incremental Linear-Time Algorithm for Recognizing Interval Graphs. *SIAM J. on Comp.*, 18(1):68–81, 1989.

[13] G.S. Lueker and K.S. Booth. A Linear Time Algorithm for Deciding Interval Graph Isomorphism. *JACM*, 26(2):183–195, 1979.

[14] R.M. McConnell and J.P. Spinrad. Construction of Probe Interval Models. In *SODA '02*, pages 866–875. ACM, 2002.

[15] T.A. McKee and F.R. McMorris. *Topics in Intersection Graph Theory*. SIAM, 1999.

[16] T. Nagoya, R. Uehara, and S. Toda. Completeness of Graph Isomorphism Problem for Bipartite Graph Classes. *IEICE Tech. Rep.*, COMP2001-93, pages 1–5, 2002.

[17] P. Zhang. Probe Interval Graphs and Its Applications to Physical Mapping of DNA. manuscript, 1994.

[18] P. Zhang. Probe Interval Graph and Its Applications to Physical Mapping of DNA. RECOMB 2000, Poster Session; available at `http://recomb2000.ims.u-tokyo.ac.jp/Posters/list-posters.html`, 2000.

[19] P. Zhang. United States Patent. Method of Mapping DNA Fragments. `http://www.cc.columbia.edu/cu/cie/techlists/patents/5667970.htm`, July 3, 2000.