COMPUTING THE BLOCKS OF A QUASI-MEDIAN GRAPH

SVEN HERRMANN AND VINCENT MOULTON

Abstract. Quasi-median graphs are a tool commonly used by evolutionary biologists to visualise the evolution of molecular sequences. As with any graph, a quasi-median graph can contain cut vertices, that is, vertices whose removal disconnect the graph. These vertices induce a decomposition of the graph into blocks, that is, maximal subgraphs which do not contain any cut vertices. Here we show that the special structure of quasi-median graphs can be used to compute their blocks without having to compute the whole graph. In particular we present an algorithm that, for a collection of \( n \) aligned sequences of length \( m \) over an alphabet of \( l \) letters, can compute the blocks of the associated quasi-median graph together with the information required to correctly connect these blocks together in run time \( O(l^2 n^2 m^2) \). Our primary motivation for presenting this algorithm is the fact that the quasi-median graph associated to a sequence alignment must contain all most parsimonious trees for the alignment, and therefore precomputing the blocks of the graph has the potential to help speed up any method for computing such trees.

1. Introduction

Quasi-median graphs are a tool commonly used by evolutionary biologists to visualise the evolution of molecular sequences, especially mitochondrial sequences (Schwarz and Dür [17]; Ayling and Brown [1]; Bandelt et al. [5]; Huson et al. [15, Chapter 9]). Their application to molecular sequence analysis was introduced for binary sequences in (Bandelt et al. [5]) and for arbitrary sequences in (Bandelt et al. [4]). A quasi-median graph can be constructed for an alignment of sequences over any alphabet [3]; for binary sequences they are also known as median graphs (Bandelt et al. [5]). An example of a quasi-median graph associated to the hypothetical alignment of sequences \( s_1 \)–\( s_9 \) is presented in Figure 1.1 (see Bandelt and Dür [3] for more details on how to construct such graphs).

Here we are interested in computing the cut vertices of a quasi-median graph as well as an associated decomposition of the graph. Recall that given a connected graph \( G = (V(G), E(G)) \), consisting of a set \( V = V(G) \) of vertices and a set \( E = E(G) \) of edges, a vertex \( v \in V \) is called a cut vertex of \( G \) if the graph obtained by deleting...
Figure 1.1. An alignment of hypothetical DNA sequences and the associated quasi-median graph. The sequences correspond to the black vertices and the columns correspond to the edges, as indicated by the labels.

v and all edges in E containing v from G is disconnected (for the basic concepts in graph theory that we use see, for example, (Diestel [9])). For example, the cut vertices of the quasi-median graph in Figure 1.1 are represented by white vertices. As with any graph, the cut vertices of a quasi-median graph decompose it into blocks, that is, maximal subgraphs which do not contain any cut vertices themselves. These blocks in turn, together with the information on how they are linked together, give rise to the block decomposition of the graph (see Section 5 for a formal definition of this decomposition that we shall use which is specific to quasi-median graphs). The main purpose of this paper is to provide an algorithm for computing the block decomposition of a quasi-median graph without having to compute the whole graph.

The results in this paper complement the well-developed theory of quasi-median networks (cf., e.g., (Bandelt et al. [8]; Imrich and Klavžar [16])). However, our primary motivation for computing the block decomposition of quasi-median graphs is provided by their close connection with most parsimonious trees (see, e.g., Felsenstein [13] for an overview of parsimony). Indeed, Bandelt and Röhlf [7] showed that the set of all most parsimonious trees for a collection of (aligned, gap-free)
sequences must be contained in the quasi-median graph of the sequences (see also Bandelt [2] for a proof of this result for median networks). More specifically, they showed that the most parsimonious trees for the sequences are in one-to-one correspondence with the Steiner trees for the sequences considered as a subset of the vertices of the quasi-median graph. It easily follows that the block decomposition of a quasi-median graph can be used to break up the computation of most parsimonious trees into subcomputations on the blocks. Of course, the quasi-median graph of an arbitrary collection of sequences may not contain any cut vertices but, as computing most parsimonious trees is NP-hard (Foulds and Graham [14]), it could still be a useful pre-processing step to compute the cut vertices of quasi-median graphs before trying to compute most parsimonious trees.

We now summarise the contents of the rest of this paper. We begin by presenting some preliminaries concerning quasi-median graphs in the next section. Then, in Section 3, we recall a characterisation of the vertices of a quasi-median graph given in (Bandelt et al. [6]), which we use in Section 4 to prove a key structural result for quasi-median graphs (Theorem 4.1). This result is a direct generalisation of Theorem 1 of (Dress et al. [6]) for median graphs, and states that the blocks in a quasi-median graph are in bijection with the connected components of a certain graph which can be associated to an alignment that captures the degree of “incompatibility” between its columns. Using this result, we also derive a characterisation of the cut vertices of a quasi-median graph (Theorem 4.6). After defining the block decomposition of a quasi-median graph in Section 5, we present our algorithm for its computation in Section 6 (Algorithm 1). In particular, we prove that this algorithm correctly computes the block decomposition (Theorem 6.1) and also show that, for a collection of $n$ aligned sequences of length $m$ over an alphabet with $l$ letters, the algorithm’s run time is $O(l^2n^2m^2)$ (Theorem 6.3). We have implemented the algorithm and it is available for download at http://www.uea.ac.uk/cmp/research/cmpbio/quasidec.

2. Preliminaries

In the following we shall define quasi-median networks in terms of partitions rather than sequences, as explained in [3]. It is quite natural to do this since, given a multiple sequence alignment as in Figure 1.1 each column of the alignment gives rise to a partition of the set of sequences in which all those sequences having the same nucleotide in the column are grouped together (note that columns with only one nucleotide are usually ignored). In particular, by also recording the number of columns giving rise to a specific partition, alignments can be recoded in terms of sets of partitions of the sequences. This whole process is described in more detail in, for example, (Bandelt and Dür [3]).

We now recall how quasi-median networks can be defined in terms of partitions. For the rest of this paper let $X$ denote an arbitrary, non-empty finite set. A partition $P$ of the set $X$ is a collection of non-empty subsets of $X$ whose union is $X$.
Example 2.1. Consider the set $X = \{s_1, s_2, \ldots, s_{10}\}$ of sequences given in Figure 1.1. The columns labelled $1, \ldots, 12$ give rise to the partitions $P_1, P_2, \ldots, P_{12}$ of $X$, respectively. For example,

$$P_1 = \{\{s_1, s_2, s_7, s_8\}, \{s_3, s_4, s_5, s_6, s_9, s_{10}\}\},$$

$$P_6 = \{\{s_1, s_2, s_3, s_4, s_7, s_8\}, \{s_5\}, \{s_6, s_{10}\}\}$$

and the element of $P_7$ containing $s_6$ is given by

$$P_7(s_6) = \{s_1, s_2, s_5, s_6, s_7, s_8, s_{10}\}.$$

Let $\mathcal{P}$ be an arbitrary set of partitions of $X$, also called a partition system on $X$. A $\mathcal{P}$-map is a map $v : \mathcal{P} \to P(X)$ that maps every partition in $\mathcal{P}$ to one of its parts. Note that, given any $x \in X$, the map $v_x : \mathcal{P} \to 2^X$ given by setting $v_x(P) = P(x)$ for $P \in \mathcal{P}$ is a $\mathcal{P}$-map. In particular, we obtain a map $\pi : x \mapsto v_x$ from $X$ to the set of all possible $\mathcal{P}$-maps.

Now, given any three $\mathcal{P}$-maps $v_1, v_2, v_3$, the quasi-median $q(v_1, v_2, v_3)$ is defined to be the $\mathcal{P}$-map

$$P \mapsto \begin{cases} v_2(P), & \text{if } v_2(P) = v_3(P), \\ v_1(P), & \text{otherwise} \end{cases}$$

for $P \in \mathcal{P}$. The quasi-median hull $H(\Phi)$ of a set $\Phi$ of $\mathcal{P}$-maps is the smallest set of $\mathcal{P}$-maps closed under taking quasi-medians, or, more formally, $H(\Phi) = \bigcup_{i \geq 0} H_i(\Phi)$, where

$$H_0(\Phi) = \Phi \quad \text{and} \quad H_i(\Phi) = \{q(v_1, v_2, v_3) \mid v_1, v_2, v_3 \in H_{i-1}(\Phi)\}.$$

The quasi-median graph $Q(\mathcal{P})$ of a partition system $\mathcal{P}$ on $X$ has vertex set $H(\pi(X))$ and edge set consisting of all those pairs $\{v_1, v_2\}$ of $\mathcal{P}$-maps in $H(\pi(X))$ that differ on precisely one partition, that is, $|\{P \in \mathcal{P} \mid v_1(P) \neq v_2(P)\}| = 1$.

Example 2.2. The quasi-median graph of the partition system described in Example 2.1 is depicted in Figure 1.1; the map $\pi$ gives the labelling of the black vertices in the graph by the sequences $s_1$ to $s_{10}$. For example, the vertex $c_4$ maps partition $P_3$ to $\{s_1, s_5, s_6, s_{10}\}$ and partition $P_6$ to $\{s_1, s_2, s_3, s_4, s_7, s_8, s_{10}\}$.

3. Strong Compatibility and Quasi-Median Graphs

We now consider a concept that is useful for understanding the structure of quasi-median graphs (cf. (Bandelt et al. [6])). Two partitions $P, Q$ of $X$ are called strongly compatible if either $P = Q$ or there exist $A \in P, B \in Q$ such that $A \cup B = X$ (see [11, p.3]). Obviously, if distinct partitions $P, Q$ of $X$ are strongly compatible, then the sets $A$ and $B$ are necessarily unique; we set $B(P, Q) = A$ and $B(Q, P) = B$. The following observation concerning these sets will be useful later.
Lemma 3.1. Let $P, Q, R$ be distinct partitions of a set $X$ such that $P$ and $Q$ are not strongly compatible and $P, Q$ are both strongly compatible with $R$. Then $B(R, P) = B(R, Q)$.

Proof. Since $R$ and $P$ are strongly compatible, we have $B(R, P) \cup B(P, R) = X$. If $B(R, Q) \neq B(R, P)$, this implies $B(R, Q) \subseteq B(P, R)$. So we get $B(Q, R) \cup B(R, Q) \supseteq B(Q, R)$, a contradiction to $P$ and $Q$ not being strongly compatible. □

A partition system $\mathcal{P}$ on $X$ is called strongly compatible if each $P, Q \in \mathcal{P}$ are strongly compatible. The following result, which is shown in the proof of [10, Lemma 3.1], will be useful later on for obtaining bounds on the number of cut vertices in a quasi-median graph.

Proposition 3.2. Let $X$ be a set of cardinality $n \geq 2$ and $\mathcal{P}$ be a strongly compatible set of partitions of $X$. Then $|\mathcal{P}| \leq 3n - 5$.

We now consider a graph that will be key for our description of the block decomposition of a quasi-median graph. The non-strong-compatibility graph for a partition system $\mathcal{P}$ on $X$ (Bandelt and Dür [3]) is the graph with vertex set $\mathcal{P}$ and edge set

$\{\{P, Q\} \mid P$ and $Q$ are not strongly compatible$\}$.

Properties of this graph have also been considered in (Schwarz and Dür [17]).

Example 3.3. We continue Example 2.1. The non-strong-compatibility graph of the partition system is depicted in Figure 3.1. For example, the partitions $P_1$ and $P_5$ are strongly compatible with $B(P_1, P_5) = \{s_3, s_4, s_5, s_6, s_9, s_{10}\}$, $B(P_5, P_1) = \{s_1, s_2, s_3, s_4, s_7, s_8, s_9, s_{10}\}$. Similarly, $P_1$ and $P_6$ are strongly compatible and – as required by Lemma 3.1 – $B(P_1, P_6) = B(P_1, P_5)$. On the other hand, $P_3$ and $P_8$ are not strongly compatible, as we cannot find elements of the partitions whose union is $X$, which gives the edge $\{3, 8\}$ in the non-strong-compatibility graph.

![Figure 3.1. The non-strong-compatibility graph for the set of partitions in Example 2.1. A vertex labelled $i$ corresponds to partition $P_i$, $1 \leq i \leq 12$.](image)

We now present some useful links between strong compatibility and quasi-median graphs. The following result was proved in ([6, Theorem 1]).

Theorem 3.4. Let $\mathcal{P}$ be a set of partitions of $X$. Then a $\mathcal{P}$-map $\varphi$ is a vertex of the quasi-median graph $Q(\mathcal{P})$ if and only if for every pair of distinct, strongly compatible partitions $P_1, P_2 \in \mathcal{P}$ either $\varphi(P_1) = B(P_1, P_2)$ or $\varphi(P_2) = B(P_2, P_1)$. 
Denote the complete graph on \(n\) vertices by \(K_n\), and, for two graphs \(G, H\), let \(G \square H\) denote the (Cartesian) product of \(G\) and \(H\), that is, the graph with vertex set \(V(G) \times V(H)\) and edge set \(\{(u, v), (u, w)\} \cup \{(v, w)\} | \{u, v\} \in E(G)\) \(\cup \{(u, w), (v, w)\} | \{u, v\} \in E(G)\). In the extreme case of pairwise strong-compatibility and non strong-compatibility for a set of partitions, we have the following descriptions of the quasi-median graph (see [6, Theorem 2, Corollary 1]).

**Theorem 3.5.** Let \(P\) be a set of partitions of \(X\). Then

(i) If every pair \(P, Q \in P\) is strongly compatible, then \(Q(P)\) is a block graph, that is, every block in \(Q(P)\) is isomorphic to a complete graph.

(ii) If no distinct \(P, Q \in P\) are strongly compatible, then \(Q(P)\) is isomorphic to \(\square_{P \in P} K_{|P|}\).

4. Cut vertices and blocks in quasi-median graphs

We now turn to understanding the cut-vertices and blocks of a quasi-median graph. By definition, for each edge \(e = \{v_1, v_2\}\) of the quasi-median graph of a set of partitions \(\mathcal{P}\) of \(X\), there exists exactly one \(P \in \mathcal{P}\) such that \(v_1(P) \neq v_2(P)\). We say that \(P\) is the partition corresponding to \(e\). Given a block \(B\) of \(Q(\mathcal{P})\) we denote by \(\mathcal{P}(B)\) the set of all \(P \in \mathcal{P}\) that correspond to some edge of \(B\). The following result that relates the connected components of the non-strong-compatibility graph of \(\mathcal{P}\) with the blocks of \(Q(\mathcal{P})\) is the key component to all that follows. Note that it has been proved in the special case where all partitions in \(\mathcal{P}(B)\) have cardinality two in [12].

**Theorem 4.1.** Let \(X\) be a finite set and \(\mathcal{P}\) be a partition system on \(X\). Then the blocks of the quasi-median graph of \(\mathcal{P}\) are in bijection with the connected components of the non-strong-compatibility graph of \(\mathcal{P}\). More specifically, a bijection is given by mapping each block \(B\) of \(Q(\mathcal{P})\) to the (necessarily) connected component of the non-strong-compatibility graph whose vertex set equals \(\mathcal{P}(B)\).

**Proof.** We prove the theorem by induction on \(|\mathcal{P}|\), the base case \(|\mathcal{P}| = 1\) being obvious.

Choose some \(P \in \mathcal{P}\) and set \(\mathcal{P}' = \mathcal{P} \setminus \{P\}\). By the induction hypothesis, the blocks of \(Q(\mathcal{P}')\) are in bijection with the connected components of the non-strong-compatibility graph of \(\mathcal{P}'\). First suppose that \(\mathcal{P}'\) is strongly compatible to all \(P' \in \mathcal{P}'\). Obviously, the non-strong-compatibility graph of \(\mathcal{P}\) is derived from the non-strong-compatibility graph of \(\mathcal{P}'\) by adding the isolated vertex \(P\). By Theorem 3.4, the vertices of \(Q(\mathcal{P})\) are either just vertices of the subgraph isomorphic to \(Q(\mathcal{P}')\) or those \(P\)-maps \(v\) defined by

\[
v(Q) = \begin{cases} B(Q, P), & \text{if } Q \in \mathcal{P}', \\ A, & \text{otherwise,} \end{cases}
\]
for some $A \in P$. There can be only one vertex which is of both types, and this is the cut vertex separating the two types of vertices and hence the new block where all edges correspond to $P$ from the other blocks. The existence of the bijection now follows from the induction hypothesis.

Now suppose $P$ is not strongly compatible to some $Q \in \mathcal{P}'$. It follows from Theorem [3.5] (ii) that all edges corresponding to $P$ and $Q$ must be in the same block. Hence, all blocks of $Q(\mathcal{P}')$ containing partitions not strongly compatible to $P$ are joined together to a new block also containing $P$. The same happens for the non-strong-compatibility graph, yielding the result. 

Example 4.2. Considering Example [2.1] we see that the non-strong-compatibility graph in Figure [3.1] has eight connected components: One whose vertex set consists of the partitions $P_1, P_2, P_3$ and $P_8$, one containing the partitions $P_5$ and $P_6$, and six isolated vertices corresponding to the remaining partitions. This is in accordance to the eight blocks of the quasi-median graph in Figure [1.1], these being the large block in the middle of the graph, corresponding to $P_1, P_2, P_3$ and $P_8$, the block on the left isomorphic to the Cartesian product of an edge and a triangle, corresponding to $P_5$ and $P_6$, two triangular blocks corresponding to the partitions $P_7$ and $P_{12}$ each having three parts, and five edges corresponding to partitions $P_4, P_9, P_{10}$ and $P_{11}$ each having two parts.

It follows from Theorem [4.1] that the collection of sets $\mathcal{P}(B)$ over all blocks $B$ of $Q(\mathcal{P})$ defines a partition $\text{Part}(\mathcal{P})$ of $\mathcal{P}$, and that the following result holds that will be useful later.

Corollary 4.3. Let $\mathcal{P}$ be a partition system of $X$ with $|\mathcal{P}| > 1$, $P \in \mathcal{P}$, $\mathcal{P}' := \mathcal{P} \setminus \{P\}$ and $I(\mathcal{P}', P) := \{Q \in \mathcal{P}' | Q$ not strongly compatible to $P\}$. Then we have

$$\text{Part}(\mathcal{P}) = \{R \in \text{Part}(\mathcal{P}') | I(R, P) = \emptyset\} \cup \left\{\bigcup \{R \in \text{Part}(\mathcal{P}') | I(R, P) \neq \emptyset\} \cup \{P\}\right\}.$$  

In particular, if $I(\mathcal{P}', P) = \emptyset$, we have $\text{Part}(\mathcal{P}) = \text{Part}(\mathcal{P}') \cup \{\{P\}\}$.

Also, by Theorem 4.1 and Proposition 3.2, the following bounds on the number of cut vertices and blocks in a quasi-median graph must hold; this will be useful for establishing run time bounds for our main algorithm.

Corollary 4.4. Let $X$ be a set of cardinality $n \geq 2$ and $\mathcal{P}$ be a set of partitions of $X$. Then $Q(\mathcal{P})$ has at most $3n - 5$ blocks and at most $3n - 6$ cut vertices.

We conclude this section by presenting a characterisation for the cut vertices in a quasi-median graph that is of independent interest, and will not be used later. First we prove a useful observation.

Lemma 4.5. Let $\mathcal{P}$ be a partition system of $X$ and $v$ a cut vertex of $Q(\mathcal{P})$. Suppose that $P_1, P_2 \in \mathcal{P}$ are distinct and that $P_i$ corresponds to an edge in the subgraph induced by $Q(\mathcal{P})$ on the set $V(C_i) \cup \{v\}$, $i = 1, 2$, where $C_1, C_2$ are two distinct connected components of the graph $Q(\mathcal{P})$ with $v$ removed. Then $P_1, P_2$ are strongly compatible, and $v(P_1) = B(P_1, P_2)$, $v(P_2) = B(P_2, P_1)$ both hold.
Proof. Since $P_1, P_2$ must be contained in distinct blocks of $Q(\mathcal{P})$, it immediately follows by Theorem 4.1 that $P_1$ and $P_2$ are strongly compatible. Moreover, by Theorem 3.4 we can assume without loss of generality that $v(P_1) = B(P_1, P_2)$. Let $\{w, w'\}$ be an edge in $Q(\mathcal{P})$ that corresponds to $P_1$. Without loss of generality, we can assume that there is a path in $Q(\mathcal{P})$ from $w$ to $v$ such that no edge in this path corresponds to $P_1$ or $P_2$. In particular, we have $w(P_1) = v(P_1) = B(P_1, P_2)$. Moreover, $w'(P_1) \neq B(P_1, P_2)$ and so by Theorem 3.4 $w'(P_2) = B(P_2, P_1)$. But, $w'(P_2) = v(P_2)$ as, by Theorem 4.1, the block containing all edges corresponding to $P_2$ must be contained in the subgraph induced by $Q(\mathcal{P})$ on the set $V(C_2) \cup \{v\}$. This completes the proof of the lemma. \hfill \Box

We now present the aforementioned characterisation of cut vertices. Note that it generalises a characterisation of cut vertices in median graphs given in [12].

**Theorem 4.6.** Let $\mathcal{P}$ be a partition system of $X$ and $v$ be a vertex of $Q(\mathcal{P})$. Then $v$ is a cut vertex of $Q(\mathcal{P})$ if and only if the graph $G_v$ with vertex set $\mathcal{P}$ and edge set $\{|P, Q|P, Q \in \mathcal{P}, P \neq Q \text{ and } v(P) \cup v(Q) \neq X\}$ is disconnected.

**Proof.** Suppose that $v$ is a cut vertex of $Q(\mathcal{P})$. Then it follows immediately by Theorem 4.1 and Lemma 4.5 that $G_v$ is disconnected.

Conversely, suppose that $G_v$ is disconnected, and, for contradiction, that $v$ is not a cut vertex of $Q(\mathcal{P})$. Note that the non-strong compatibility graph of $\mathcal{P}$ is a subgraph of $G_v$. Hence the non-strong compatibility graph of $\mathcal{P}$ is disconnected. Therefore, by Theorem 4.1 there are at least two blocks in $Q(\mathcal{P})$.

Now, suppose $B$ is the block of $Q(\mathcal{P})$ containing $v$. By Theorem 4.1 there must exist some block $B' \neq B$ of $Q(\mathcal{P})$ such that $\mathcal{P}(B')$ is contained in the vertex set of some connected component of $G_v$ that is not equal to the connected component of $G_v$ whose vertex contains $\mathcal{P}(B)$. Let $w$ be the cut vertex of $Q(\mathcal{P})$ contained in $B$ which lies on a shortest path from $v$ to some vertex in $B'$. Let $P \in \mathcal{P}$ correspond to the edge on this path incident with $w$ (which must exist as $v$ is not a cut vertex), and let $P' \in \mathcal{P}(B')$. Then, by Lemma 4.5 $w(P) = B(P, P')$ and $w(P') = B(P', P)$. Moreover, by Theorem 4.1 $w(P') = v(P')$ and $w(P) \neq v(P)$. Hence $v(P) \cup v(P') \neq X$, which is a contradiction as $P$ and $P'$ are in distinct components of $G_v$. \hfill \Box

5. The Block Decomposition of a Quasi-Median Graph

As stated in the introduction, we want to determine the blocks of the quasi-median graph $Q(\mathcal{P})$ of a partition system $\mathcal{P}$ without having to compute $Q(\mathcal{P})$ itself. To do this, rather than computing the blocks of $Q(\mathcal{P})$ directly, we shall compute some sets associated with each block which we now define.

Given a block $B$ of $Q(\mathcal{P})$, we let $X(B) = V(B) \cap \pi(X)$ denote the set of vertices in $B$ labelled by elements in $X$. $\mathcal{P}(B)$ the set of partitions in $\mathcal{P}$ corresponding to edges of $B$ and $S(B)$ the set of cut vertices of $Q(\mathcal{P})$ that are in $B$ but not in $X(B)$. Note that $X(B)$ or $S(B)$ can be empty, but that $X(B) \cup S(B)$ is never empty. We will also
consider the set $\mathcal{P}_r(B)$ of partitions of the set $X(B) \cup S(B)$ that is induced by, for each $P \in \mathcal{P}(B)$, removing all those edges in $B$ that correspond to $P$.

**Example 5.1.** For the large block $B$ in the middle of the quasi-median graph in Example [2.1], we have $X(B) = \{s_7, s_8\}$, $S(B) = \{e_1, e_4, e_5\}$, $\mathcal{P}(B) = \{P_1, P_2, P_3, P_8\}$ and $\mathcal{P}_r(B) = \{P_1', P_2', P_3', P_8'\}$, where

\[
\begin{align*}
P_1' &= \{\{s_7, s_8, e_5\}, \{e_1, e_4\}\}, & P_2' &= \{\{e_1, e_5, s_8\}, \{s_7, e_4\}\}, \\
P_3' &= \{\{s_7, s_8, e_1\}, \{e_4, e_5\}\}, & P_8' &= \{\{s_7, e_1, e_4\}, \{s_8, e_5\}\}.
\end{align*}
\]

Now, we define the **block decomposition** $\mathcal{B}(\mathcal{P})$ of the quasi-median graph of a partition system $\mathcal{P}$ on the set $X$ to be the set

$$\{(X(B), S(B), \mathcal{P}_r(B)) \mid B \text{ is a block of } Q(\mathcal{P})\}.$$ 

Our main aim is to compute this decomposition without having to compute $Q(\mathcal{P})$. Note that in view of the following lemma we can always reconstruct $Q(\mathcal{P})$ from $\mathcal{B}(\mathcal{P})$.

**Lemma 5.2.** Given a partition system $\mathcal{P}$ and a block $B$ of $Q(\mathcal{P})$, the quasi-median graph $Q(\mathcal{P}_r(B))$ is isomorphic to $B$.

**Proof.** By definition, a $\mathcal{P}$-map $v$ is a vertex of the block $B$ if and only if $v$ is contained in some edge of $Q(\mathcal{P})$ corresponding to an element of $\mathcal{P}(B)$. Consider now the $\mathcal{P}_r(B)$-map $v'$ that maps a partition $P' \in \mathcal{P}_r(B)$ to the same part that $v$ maps the partition $P$. This is a vertex of $Q(\mathcal{P}_r(B))$ and it can be easily seen that the map $v \mapsto v'$ induces the desired isomorphism between $B$ and $Q(\mathcal{P}_r(B))$. \qed

**Remark 5.3.** In [17, Theorem 3], Schwarz and Dür define what they call the **Block Decomposition of a Quasi-Median Network**. However, they do not use the notion of **block** in the usual graph theoretical way. Instead, they work with a notion that is suitable for their aim of visualising quasi-median graphs. In particular, their blocks depend on an arbitrary vertex of the quasi-median graph which can be chosen in a suitable way to obtain improved visualisations.

In what follows, we shall not directly compute the the block decomposition of $Q(\mathcal{P})$, but instead some closely related data from which the block decomposition can be easily computed.

To this end, let $S(\mathcal{P})$ denote the union of all $S(B)$ with $B$ a block of $Q(\mathcal{P})$; we call any element in $S(\mathcal{P})$ an **extra vertex**. For $v \in S(\mathcal{P})$ we denote the set of all blocks $B$ in $Q(\mathcal{P})$ with $v \in S(B)$ by $B(v)$. An element $x \in X$ is **in the direction** of $B$ with respect to $v \in S(B)$ if every path from $x$ to $v$ has an edge in $B$. Note that since all vertices of $Q(\mathcal{P})$ are elements of the quasi-median hull of $\pi(X)$, there always exists such an element $x(v, B)$ although this element is not necessarily unique.

**Lemma 5.4.** Suppose that $\mathcal{P}$ is a partition system on $X$ and $B$ is a block of $Q(\mathcal{P})$. If we are given the sets $X(B)$, $S(B)$, $\mathcal{P}(B)$ and, for each $v \in S(B)$, some element
x(v, B) in the direction of B with respect to v, then we can obtain the set \( \mathcal{P}_r(B) \) from the set \( \mathcal{P}(B) \) in time \( O(nm) \), where \( n = |X| \), \( m = |\mathcal{P}| \).

Proof. For each partition \( P \in \mathcal{P}(B) \) we construct a partition \( P' \) of \( X(B) \cup S(B) \) as follows. Elements of \( X(B) \) are in that part of \( P' \) that they are in \( P \). For each \( v \in S(B) \) we choose some \( C \in B(v) \setminus \{B\} \) and put \( v \) in that part of \( P' \) that \( x(v, B) \) is in \( P \). Repeating this for all partitions \( P \in \mathcal{P}(B) \) gives us the set \( \mathcal{P}_r(B) \). This procedure can be carried out in time \( O(nm) \), giving the desired run time bound. \( \square \)

Example 5.5. To compute \( \mathcal{P}_r(B) \) from \( \mathcal{P}(B) \) and the information \( x(v, B) \) for all \( v \in S(B) \) for the block \( B \) in Example 5.1, assume that \( x(e_1, B_7) = s_3 \), \( x(e_4, B_4) = s_5 \) and \( x(e_5, B_{10}) = s_1 \), where, for this moment, we denote by \( B_i \) the block containing the (sole) partition \( P_i \).

Now, we start out with partition \( P_1 \) and have to check in which part of the partition the extra points \( e_1, e_4 \) and \( e_5 \) are contained. Since \( x(e_1, B_7) = s_3 \), we substitute \( s_3 \) for \( e_1 \) in \( P_1 \) and, similarly, we substitute \( s_5 \) for \( e_4 \) and \( s_1 \) for \( e_5 \). Deleting all \( x \in X \setminus X(B) \) in the remaining partition yields the partition \( P'_1 \). After performing the same process for \( P_2, P_3 \) and \( P_8 \), we obtain the set \( \mathcal{P}_r(B) \).

So, to compute the block decomposition of the quasi-median graph of a partition system \( \mathcal{P} \) it suffices to compute, for each block \( B \) of \( Q(\mathcal{P}) \), the sets \( X(B) \), \( S(B) \) and \( \mathcal{P}(B) \), and also, for each \( v \in S(B) \) and \( B \in B(v) \), some element \( x(v, B) \) in the direction of \( B \) with respect to \( v \). In the next section we shall present an algorithm for doing precisely this.

6. Computing the Block Decomposition of a Quasi-Median Graph

We now present our approach to computing the block decomposition of a partition system \( \mathcal{P} \) following the strategy presented at the end of the last section. We start with the block decomposition of an empty set of partitions on \( X \) (which is itself empty) and iteratively add each \( P \in \mathcal{P} \) to build up the decomposition. In particular, at each stage, for each block \( B \) (either existing or new) we compute the sets \( X(B), S(B), \mathcal{P}(B) \), together with elements \( x(v, B), v \in S(B), B \in B(v) \). To do this we use Algorithm 1, the main elements for which are as follows.

First, for each given block \( B \), we check whether or not there exists some partition in \( \mathcal{P}(B) \) that is not strongly compatible to the newly added partition \( \mathcal{P} \) and thereby also compute which elements of \( X \) must be added to our new block. This is done in the function is_compatible described in Algorithm 2. This function returns TRUE if the new partition \( P \) is compatible to all partitions \( Q \) in the block \( B \). All blocks \( B \) with is_compatible(\( P, B \))=TRUE remain blocks for the new block decomposition, and all other blocks are joined (together with \( P \)) to form a new block that is added to the decomposition. This is done in the function join_blocks outlined in Algorithm 3.

We now prove that this approach really works:
Algorithm 1: Algorithm to add a partition.

**Input:** The set $\mathcal{B} = \{(X(B), S(B), P(B)) : B \text{ a block of } Q(\mathcal{P})\}$ for a partition system $\mathcal{P}$ and, for each $v \in S(B)$ and $B \in B(v)$, some element $x(v, B)$ in the direction of $B$ with respect to $v$, together with some partition $P \notin \mathcal{P}$.

**Output:** The same data for $\mathcal{P} \cup \{P\}$.

1. Create a new block $C$ with $X(C) = X$, $S(C) = \emptyset$, $P(C) = \{P\}$;
2. Create a new extra vertex $v$;
3. $\mathcal{B}_{\text{incomp}} \leftarrow \emptyset$;
4. foreach $B \in \mathcal{B}$ do
   5. if $\neg \text{is_compatible}(P, B)$ then
      6. Add $B$ to $\mathcal{B}_{\text{incomp}}$;
   7. end
   8. else
      9. Choose some $Q \in P(B)$;
     10. $X(B) \leftarrow X(B) \cap B(P, Q)$;
     11. if $X(B) \notin B(P, Q)$ and $X(C) \cap X(B) = \emptyset$ then
        12. Choose some $x \in B(P, Q)$;
        13. if There exists some $v \in S(C)$ such that $x(v, B)$ and $x$ are in the same part of $P$ then
           14. $w \leftarrow v$;
        15. end
        16. else
           17. $w \leftarrow \text{new extra vertex}$;
        18. end
        19. $x(w, B) \leftarrow \text{add_extra_vertex}(w, B)$;
     20. $x(w, C) \leftarrow \text{add_extra_vertex}(w, C)$;
   21. end
5. if $\mathcal{B}_{\text{incomp}} \neq \emptyset$ then
   6. add_blocks($C, \mathcal{B}_{\text{incomp}}$);
5. end
27. return $\mathcal{B} \cup \{(X(C), S(C), P(C))\}$ and the elements $x(w, C)$;

**Theorem 6.1.** Algorithm 1 is correct.

**Proof.** We first show that if the sets $P(B)$ and $X(B)$ have been correctly computed for all blocks $B$ of the quasi-median graph of the partition system $\mathcal{P} \setminus \{P\}$, then they are correctly computed for $\mathcal{P}$. 
To see that \( \mathcal{P}(B) \) is correct, note that the set \( \mathcal{P}(C) \) for the new block \( C \) is initialised as \( \{ P \} \) and in the function `add_blocks` all partitions of blocks containing partitions incompatible to \( P \) are added and the corresponding blocks deleted. Hence, it follows from Corollary 4.3 that \( \mathcal{P}(B) \) is correct for all blocks of \( \mathcal{Q}(P) \).

We now turn to the correctness of \( X(B) \). Consider first a block \( B \) for which every partition \( Q \in \mathcal{P}(B) \) is strongly compatible with \( P \). The elements of in \( X(B) \) stay in \( X(B) \) if they are in \( B(Q, P) \), and similarly move to \( X(C) \) for the new block \( C \) if they are in \( B(Q, P) \). But, by Theorem 3.5 (i), the quasi-median graph \( \mathcal{Q}(\{ P, Q \}) \) has two blocks \( B, C \) with \( \mathcal{P}(B) = \{ P \}, X(B) = B(Q, P) \) and \( \mathcal{P}(C) = \{ Q \}, X(Q) = B(P, Q) \). It follows that \( X(B) \) is correct. Otherwise, if some \( Q \in \mathcal{P}(B) \) is not compatible to \( P \), then the corresponding block is deleted and all elements are simply joined to those in \( X(C) \), as required. So, using a similar argument for \( \mathcal{Q}(\{ P, Q \}) \), it follows that \( X(C) \) is also correct.

It remains to show that the blocks are added in a proper way, that is, all of the extra vertices are contained in the blocks that they really belong to. This is taken care of by the condition in Line 11 of Algorithm 1: There is no need to add extra vertices for adding two blocks if they already share an element of \( X \) and having \( X(B) \nsubseteq B(P, Q) \) ensures that blocks are only added if needed. Moreover, Algorithm 4 ensures that elements in the direction of some block are computed. Indeed, suppose all existing \( x(\cdot, \cdot) \) are correct. To see that Algorithm 4 returns an element of \( x \) that is in the direction of \( B \) first not that if \( x \in X(B) \) and \( X(B) \neq \emptyset \), then \( x \) is clearly in the direction of \( B \) with respect to \( v \). Furthermore, every \( w \in S(B) \setminus \{ v \} \) is in the direction of \( B \) with respect to \( v \) and so every element in the direction of any \( C \in B(w) \setminus \{ B \} \) with respect to \( w \) is in the direction of \( B \) with respect to \( v \). This completes the proof of the theorem.

\[ \square \]

\textbf{Algorithm 2: } Check if a partition is compatible with all partitions arising from a block.

1 \textbf{i}s\_c\textbf{ompatible}(\textbf{P}, \textbf{B})
2 \hspace{1em} \textbf{foreach} Partition \textbf{Q} \in \textbf{P}(\textbf{B}) \textbf{do}
3 \hspace{2em} if \textbf{P} and \textbf{Q} are strongly compatible then
4 \hspace{3em} \textbf{X}(\textbf{C}) \leftarrow \textbf{X}(\textbf{C}) \cap B(\textbf{Q}, \textbf{P});
5 \hspace{2em} end
6 \hspace{2em} else
7 \hspace{3em} return FALSE;
8 \hspace{2em} end
9 \hspace{1em} end
10 return TRUE;

We conclude with an analysis of the run time of Algorithm 1. First, we compute the time needed to check whether two partitions are compatible.
Algorithm 3: Add all blocks incompatible to $P$.

1. add_blocks($C, \mathcal{B}_{\text{incomp}}$)
2. $X(C) \leftarrow \emptyset$
3. foreach $w \in S(C)$ do
   4. if $B(w) \cup \mathcal{B}_{\text{incomp}} = \emptyset$ then
      5. Delete the extra vertex $w$ from $S(C)$;
   6. end
7. end
8. foreach $B \in \mathcal{B}_{\text{incomp}}$ do
   9. Remove $B$ from $\mathcal{B}$;
10. $X(C) \leftarrow X(C) \cup X(B)$;
11. $\mathcal{P}(C) \leftarrow \mathcal{P}(C) \cup \mathcal{P}(B)$;
12. foreach $w \in S(B)$ do
    13. Add $w$ to $S(C)$;
14. $x(w, C) \leftarrow x(w, B)$;
15. end
16. end

Algorithm 4: Add an extra vertex to a block.

1. add_extra_vertex($v, B$)
2. Add $v$ to $S(B)$;
3. if $X(B) \neq \emptyset$ then
   4. Choose some $x \in X(B)$;
   5. return $x$;
end
7. Choose some $w \in S(B) \setminus \{v\}$;
8. Choose some $C \in B(w) \setminus \{B\}$;
9. return $x(w, C)$;

Lemma 6.2. Let $P$ and $Q$ be partitions of $X$ with $|P| = k_1$, $|Q| = k_2$ and $|X| = n$. Then checking strong compatibility and computing $B(P, Q)$ and $B(Q, P)$ in case they are compatible can be done in time $O(k_1 k_2 n)$.

Proof. For each $A \in P$ and $B \in Q$ we check if the $A \cup B = X$. If this is the case, $P$ and $Q$ are compatible and $B(P, Q) = B$, $B(Q, P) = A$. Such a check can be done in linear time in $n$ and there are $k_1 \cdot k_2$ such pairs. \qed

Theorem 6.3. The algorithm computes the block decomposition of a partition system $\mathcal{P}$ on $X$ in time $O(k^2 n^2 m^2)$, where $n = |X|$, $m = |\mathcal{P}|$ and $k = \max(|P| \mid P \in \mathcal{P})$. 
Proof. We claim that Algorithm 1 runs in time $O(k^2n^2m)$. Since this algorithm is executed once for each partition, the theorem then follows by Lemma 5.4 and Corollary 4.4.

It follows from Lemma 6.2 that the function \texttt{is_compatible} in Algorithm 2 runs in time $O(k^2n \cdot |\mathcal{P}(B)|)$. The rest of the first loop in Algorithm 1 is dominated by the condition in Line 13. However, since the number of extra vertices of $Q(P)$ is linear in $n$ by Proposition 4.4, this test can be performed in $O(n^2)$. Since each partition can only be in one block, this shows that the loop in Algorithm 1 needs $O((k^2n + n^2)m)$ time. For the function \texttt{add_blocks} the run time of the first loop is bound by $O(n^2)$, taking into account that by Proposition 4.4 the number of extra vertices and the number of blocks are linear in $n$. The same holds for the second loop, so \texttt{add_blocks} runs in time $O(n^2)$. Altogether, we get that Algorithm 1 runs in time $O((k^2n + n^2)m + n^2 + kn^2) = O(k^2n^2m)$, as claimed. \qed

Note that, translated into the language of sequences used in the introduction, this results implies that the block decomposition of the quasi-median graph of $n$ aligned sequences of length $m$ over an alphabet with $l$ characters can be computed in time $O(l^2n^2m^2)$, as the size of each corresponding partition, and hence $k$, is bounded by $l$.

REFERENCES

1. Sarah C. Ayling and Terence A. Brown, Novel methodology for construction and pruning of quasi-median networks, BMC Bioinformatics 9 (2008), 10 pp.
2. Hans-Jürgen Bandelt, Median hulls as steiner hulls in rectilinear and molecular sequence spaces, Graph-Theoretic Concepts in Computer Science (Andreas Brandstädter and Van Le, eds.), Lecture Notes in Computer Science, vol. 2204, Springer Berlin / Heidelberg, 2001, pp. 1–7.
3. Hans-Jürgen Bandelt and Arne Dür, Translating dna data tables into quasi-median networks for parsimony analysis and error detection., Mol Phylogenet Evol 42 (2007), no. 1, 256–71.
4. Hans-Jürgen Bandelt, Peter Forster, and Arne Röhl, Median-joining networks for inferring intraspecific phylogenies., Molecular Biology and Evolution 16 (1999), no. 1, 37–48.
5. Hans-Jürgen Bandelt, Peter Forster, Bryan Sykes, and Martin B. Richards, Mitochondrial portraits of human populations using median networks, Genetics 141 (1995), no. 2, 743–753.
6. Hans-Jürgen Bandelt, Katharina T. Huber, and Vincent Moulton, Quasi-median graphs from sets of partitions, Discrete Appl. Math. 122 (2002), no. 1-3, 23–35. MR 1907821 (2003c:05039)
7. Hans-Jürgen Bandelt and Arne Röhl, Quasi-median hulls in Hamming space are Steiner hulls, Discrete Appl. Math. 157 (2009), no. 2, 227–233. MR 2479797 (2010a:05057)
8. Hans-Jürgen Bandelt, Henry Martyn Mulder, and Elke Wilkeit, Quasi-median graphs and algebras, Journal of Graph Theory 18 (1994), no. 7, 681–703.
9. Reinhard Diestel, Graph theory, Springer-Verlag, Heidelberg, 2010.
10. Andreas Dress, Katharina T. Huber, Jacobus Koolen, Vincent Moulton, and Andreas Spiller, An algorithm for computing cutpoints in finite metric spaces, J. Classification 27 (2010), no. 2, 158–172. MR 2726316 (2011k:54035)
11. Andreas Dress, Vincent Moulton, and Michael Steel, *Trees, taxonomy, and strongly compatible multi-state characters*, Adv. in Appl. Math. 19 (1997), no. 1, 1–30. MR 1453403 (99g:92003)

12. Andreas W. M. Dress, Katharina T. Huber, Jack Koolen, and Vincent Moulton, *Blocks and cut vertices of the buneman graph*, SIAM Journal on Discrete Mathematics 25 (2011), no. 4, 1902–1919.

13. Joseph Felsenstein, *Inferring phylogenies*, Sinauer Associates, Inc., 2004.

14. Leslie R. Foulds and Ron L. Graham, *The steiner problem in phylogeny is np-complete*, Adv Appl Math 3 (1982), 43–49.

15. Daniel Huson, Regula Rupp, and Celine Scornavacca, *Phylogenetic networks. concepts, algorithms and applications*, Cambridge University Press, 2010.

16. Wilfried Imrich and Sandi Klavžar, *Product graphs*, Wiley-Interscience Series in Discrete Mathematics and Optimization, Wiley-Interscience, New York, 2000, Structure and recognition, With a foreword by Peter Winkler. MR 1788124 (2001k:05001)

17. Konrad Schwarz and Arne Dür, *Visualization of quasi-median networks*, Discrete Appl. Math. 159 (2011), no. 15, 1608–1616.

School of Computing Sciences, University of East Anglia, Norwich, NR4 7TJ, UK