

Journal of Bioinformatics and Computational Biology
© Imperial College Press

**PAIRWISE PARTIAL ORDER ALIGNMENT AS A SUPERGRAPH PROBLEM
— ALIGNING ALIGNMENTS REVISITED***

D. STOTT PARKER

*Computer Science Department
University of California
Los Angeles, California 90095-1596, USA
stott@cs.ucla.edu*

CHRISTOPHER J. LEE

*Department of Chemistry & Biochemistry
University of California
Los Angeles, California 90095-1569, USA
leec@mbi.ucla.edu*

Received (20 October 2003)
Revised (20 October 2003)
Accepted (20 October 2003)

Partial Order Alignment (POA) has been proposed recently as an alternative to conventional sequence alignment. Instead of the familiar tabular alignments, POA methods produce a partial order — a labeled directed acyclic graph — that includes the input sequences. In this paper, we formalize POA in terms of graphs, and show it corresponds to finding a Minimal Common Supergraph for a set of partial order graphs. We also show how this formulation may serve as a guide in the development of new alignment algorithms, and in addressing perennial problems, such as how to *align alignments*.

1. Introduction

Partial Order Alignment (POA)^{14,18,21,23,28} is a recently-developed alternative to traditional sequence alignment methods^{11,16}. Rather than a tabular arrangement of sequences, alignments with POA consist of a partial order (labeled directed acyclic graph) that subsumes the sequences, as shown in Figure 1.

Recently an independent evaluation by Lassman and Sonnhammer²² reported the POA approach to have real advantages over traditional alignment methods. To illustrate quickly why this would be so, and to show that POA has a number of possible applications, some examples of alignments are reproduced in Figures 2–5. The biological implications of these diagrams²¹ is omitted here; they are only meant to convey a rough impression of possibilities, and suggest that partial order structures are good models for some aspects of biological

*POA home page: <http://www.bioinformatics.ucla.edu/poa>. Work supported by NSF grant IIS 0082964. Project home page: <http://www.bioinformatics.ucla.edu/leelab/db>

2 Parker and Lee

sequences that are not as well captured by the traditional tabular arrangements. POA is of interest in its own right, since it gives a global, graph-like perspective that can be used with local perspectives afforded by dynamic programming when attacking alignment problems.

An important theme in this paper is the role of scale, or granularity, in alignment. Figures 3, 4, and 5 address aspects of sequences at successively larger granularities. That this can be done is not only an interesting aspect of POA from the standpoint of biology, but also an important aspect of POA from the standpoint of algorithmic complexity. Although some algorithms in this paper may be impractical at the granularity of nucleotides, they can be quite practical at the granularity of splices (Figure 4) or protein domains (Figure 5).

This paper attempts to formalize POA in graph-theoretic terms; set a foundation for implementations; and formulate sequence alignment questions from a POA perspective, such as how to *align alignments*¹⁹. This paper is specifically focused on *pairwise* POA. A companion paper²⁴ analyzes the generalization to multiple alignment.

We begin by considering a view of alignment as a ‘fusion’ process that integrates sequences into a model (a partial order graph). The resulting intuition about fusion is then formalized as POA. We show the pairwise POA problem to be NP-complete. As a result (or nevertheless) we then develop a generic algorithm for POA that works by enumerating sets of possible fusions, or equivalently by enumerating certain subgraphs of a derived *fusion graph*. We conclude by considering applications, a scheme for aligning alignments, and avenues for future work.

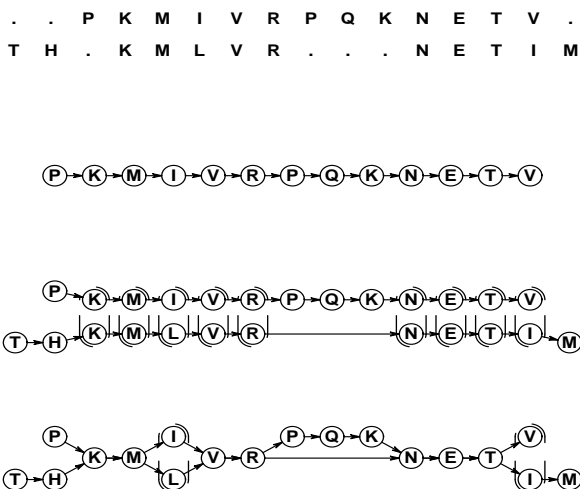


Fig. 1. Partial Order Alignment for the two input sequences *PKMIVRPQKNETV*, *THKMLVLRNETIM*. The first line gives a pairwise alignment for these sequences. The second line is a linear graph — the PO corresponding to the sequence *PKMIVRPQKNETV*. The third line (ignoring the dashed ovals) includes only the input sequences as paths, while the fourth line is a POA that includes many other sequences as paths. However, the fourth line has the minimal number of nodes among all POs that include the input sequences as paths, while the third has the maximal number (the sum of all input sequence lengths). The fourth line is thus both a POA for the two input sequences and a solution to the (pairwise) PO Alignment problem for these sequences as graphs. It can also be viewed as an sequence emission model — a HMM- or grammar-like generator — for these two sequences.

CONSENSUS	a.gttcctgc.tgcgtttgcaggactgatgtactt.gtttgtgagg.caa
Hs#S663801	a.gttcctgc.tgcgtttgcaggacttatgtactt.gtttgtgagg.caa
Hs#S337687	aagttcctgc.tgcgtttgcaggactgatgtacttggtttgnaggcaa
Hs#S629177	a.gttcctgc.tgcgtttgcaggactgatgtactt.gtttgnagg.caa
Hs#S672957	a.gttcctgc.tgcgtttgc.....
Hs#S672182	a.gttcctgc.tgcgtttgcaggactgatgtactt.gttt.....
Hs#S674099	a.gttcctgc.tgcgtttgcaggactgatgtactt.gtttgtgagg.caa
Hs#S196113	a.gtttctgn.tgngtttgcaggactgatgtactt.gtttgtgagg.caa
Hs#S994400gtacnt.gtttgtgagg.cta
Hs#S80460	a.gttcctgc.tgcgtttgcaggactgatgtactt.gtttgtgagg.caa
Hs#S1988018	a.gttcctgc.tgcttttgcaggactgatgtactt.gatttgtgagg.caa
Hs#S1794113	a.gttcctgc.tgcgcttgcaggactgatgtactt.gtttgtgagg.caa
Hs#S4698	a.gttcctgc.tgcgtttgcaggactgatgtactt.gtttgtgagg.caa
Hs#S813765	a.gt.cctgc.g.cgttgc.ggacggatgtactt.gtt.gtgagg.caa
Hs#S1184845g.caa
Hs#S1577463gg.caa
Hs#S914987ctgatgtactt.gtt.gtgaggcaa
Hs#S1985364	a.gttcctgc.tgcgtttgcaggactgatgtactt.gtttgtgagg.caa
Hs#S1465644	.gttc.tgcctgcgtttgcaggactgatgtactt.gtttagt.aag.caa
Hs#S1850471	c.gttactgc.ggggttgcaggactcatg.acttgttngt.agg.caa

Fig. 2. Sample Multiple Sequence Alignment for EST sequences from UniGene cluster Hs.100194

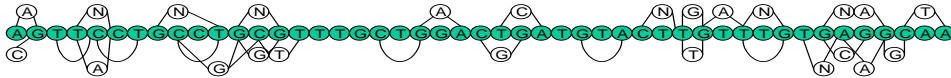


Fig. 3. POA summarizing the sequences shown in Figure 2

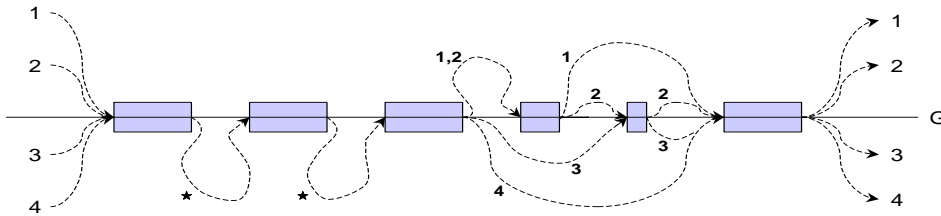


Fig. 4. Partial Order Alignment showing four different observed alternatively spliced mRNA forms, and the genomic sequence, of human gene HLA-DM β from Unigene cluster Hs.1162. Exons are shown as rectangles; asterisks indicate that all four forms share an indicated sequence.

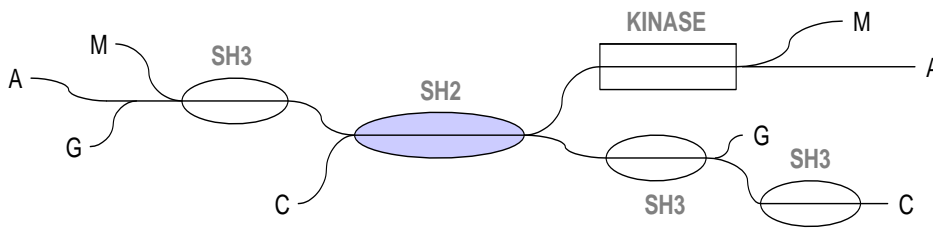


Fig. 5. POA showing recombination among the 4 multidomain protein sequences with Swissprot identifiers MATK_HUMAN (M), ABL1_HUMAN (A), GRB2_HUMAN (G), and CRKL_HUMAN (C); protein domains are shown as ovals and rectangles.

1.1. Alignment as Graph Fusion

In what follows sequences are viewed as graphs in which nodes are labeled with the letters of the sequences, as shown in Figure 1.

When sequences are represented as graphs, *alignment* of sequences corresponds naturally to *fusion* of their nodes. That is, nodes (letters) that are aligned can be integrated into a single node. Figure 1 shows a simple example, in which 7 fusions are made (for the subsequences of nodes corresponding to the letters KMVRNET).

This fusion process has the effect of integrating multiple sequence emission models into a single model. Usually there is more than one possible result. The shape of the resulting graph depends on the specific fusions that are permitted (e.g., we are permitted to fuse nodes having identical letters), and on the specific choices of fusions made.

Generally, as a Parsimony Principle, we seek a *minimal model* — a model that is as concise as possible. Intuitively, for example, the final graph in Figure 1 is a minimal model for the two sequences shown. To obtain a minimal model, we seek a set of fusions that minimizes the size of the resulting graph.

Definition 1. Each node and edge in a graph has a **size**: an associated numeric value. The **size of a graph** is the sum of the sizes of its nodes and the sizes of its edges.

This definition is general enough to cover many notions of size. It can handle statistical notions such as log-odds measures. It can also handle graph-theoretic notions such as *node+edge graph size*, discussed later, which assigns high weight to the number of nodes and lower weight to the number of edges.

Definition 2. There are two kinds of node fusion:

- **identity fusion**: fusion of identically-labeled nodes.
- **substitution fusion**: fusion of differently-labeled nodes.

The node resulting from the fusion is labeled with a set containing all labels from the fused nodes. This set is called an **align ring**¹⁴.

For example, in Figure 1, only identity fusion is shown. The two final nodes labeled with the letters I and V could be fused into a single node labeled with align ring {V,I}. Labeling nodes with sets of letters differs in a minor way from the initial POA paper²¹, which used dashed circles around nodes to denote align rings.

In this paper, alignment is fusion. Furthermore fusion is limited to a predefined set of *possible fusions*, from which we seek optimal subsets.

Definition 3. A **set of possible fusions** PF for labeled graphs $G_1 = (V_1, E_1, \ell_1)$ and $G_2 = (V_2, E_2, \ell_2)$ is a subset $PF \subseteq V_1 \times V_2$ of the cartesian product of their nodes.

Some natural sets of possible fusions include the set of all identity fusions, the set of fusions for pairs of letters for which the probability of point mutation is high (e.g., as measured in a PAM/BLOSUM matrix¹¹), or the set of fusions for nodes whose codon

triplets differ by at most a single base change (such as L ↔ l, L ↔ V, etc.¹¹).

$PF = \{ (u, v) \in V_1 \times V_2 \mid \ell_1(u) = \ell_2(v), \text{ i.e., } u \text{ and } v \text{ have identical labels} \}$

$PF = \{ (u, v) \in V_1 \times V_2 \mid \text{the } [\ell_1(u), \ell_2(v)] \text{ entry in the BLOSUM62 matrix is positive} \}$

$PF = \{ (u, v) \in V_1 \times V_2 \mid \text{the codon labels of } u, v \text{ differ by at most a single base change} \}$.

This paper focuses on problems in which the set of possible fusions PF has nontrivial structure, such as when the set of labels (alphabet) is relatively large and substitutions are held to a high standard. In trivial problems where $PF = V_1 \times V_2$ the alignment problem becomes less a graph problem, emphasizing the model structure we are interested in, and more a numerical optimization problem. The purpose of this paper is to investigate basic properties of a graph-theoretic formulation.

2. Partial Order Alignment as a Graph Problem

Definition 4. A **labeled, sized, directed graph** $G = (V, E, \ell, s)$ is a graph such that:

- V is the set of **nodes** of G .
- $E \subset V \times V$ is the set of **directed edges** of G ; each pair $(u, v) \in E$ represents the directed edge $(u \rightarrow v)$.
- **node labels** of G are defined by $\ell : V \rightarrow 2^L$, a function mapping individual nodes to nonempty subsets of L , a fixed **label set** (e.g., alphabet of individual **letters**).
- **sizes** of nodes and edges in G are defined by $s : V \cup E \rightarrow \mathfrak{R}$, a function mapping individual nodes and edges to numeric values. The **size of G** is $\sum_{x \in V \cup E} s(x)$.

Thus each node and edge in the graphs has a *size* — an associated numeric value — and the **size of a graph** is the sum of its node sizes and edge sizes. A node size $s(v)$ can, for example, depend on $\ell(v)$ or other properties of node v . Again, this definition is general enough to handle many notions of size.

For example, the **node+edge size of a graph** $G = (V, E)$ assigns every node has size 1 and every edge has size c , where c is small (e.g., $0 \leq c \leq 1/n^2$, where n is an upperbound on the number of nodes). This yields the total graph size $|V| + c|E|$, in which $|V|$ is of primary importance; $|E|$ has at most secondary influence on size. The node+edge size measure implements a lexicographic ordering on $|V|$ first and $|E|$ second.

Definition 5. Let $G_1 = (V_1, E_1, \ell_1, s_1)$ and $G_2 = (V_2, E_2, \ell_2, s_2)$ be graphs in which V_1 and V_2 are disjoint sets, and let $PF \subseteq V_1 \times V_2$ be a set of possible node fusions.

Formally, for any subset $F \subseteq PF$, the **result of performing fusions F** is a **supergraph** $G = (V, E, \ell, s)$ such that $V = (V_1 \cup V_2) \text{ mod } F$ and $E = (E_1 \cup E_2) \text{ mod } F$. Here $X \text{ mod } F$ denotes the object X upon which F has been applied as an equivalence relation; equivalent objects are **fused**. Thus G contains both G_1 and G_2 as subgraphs (mod F).

Informally, the result of performing fusions F is to make the previously disjoint sets V_1 and V_2 subsequently share a common subset V' of fused nodes. Both G_1 and G_2 are contained in the graph G that results from this fusion, and G is thus a supergraph.

The set V' of fused nodes also defines a **subgraph** $G' = (V', E', \ell, s)$ of G . Within this subgraph E' consists of the edges in E_1 and E_2 that only connect nodes in V' , i.e., $E' = E \cap (V' \times V')$. The **node label** ℓ of the node $v \in V'$ resulting from fusing $v_1 \in V_1$ and $v_2 \in V_2$ is defined by $\ell(v) = \ell_1(v_1) \cup \ell_2(v_2)$, whereas the **size** $s(v)$ reflects the sizes of v_1 and v_2 as well as some ‘score’ from fusing v_1 and v_2 . For non-fused nodes $v \in V - V'$ both $\ell(v)$ and $s(v)$ are defined by the graph containing v .

Definition 6. A **PO (Partial Order)** is a graph $G = (V, E, \ell, s)$ that is **acyclic**, i.e., there is no value $k > 1$ and no subset $\{v_{i_1}, \dots, v_{i_k}\} \subseteq V$ such that the cyclic sequence of edges $v_{i_1} \rightarrow v_{i_2}, \dots, v_{i_{k-1}} \rightarrow v_{i_k}, v_{i_k} \rightarrow v_{i_1}$ is a subset of E .

Every **sequence** $a_1 \dots a_n$ naturally defines a PO: a graph $G = (V, E, \ell, s)$ that has nodes $V = \{v_1, \dots, v_n\}$, edges $E = \{(v_1, v_2), (v_2, v_3), \dots, (v_{n-1}, v_n)\}$, and the node labels $\ell(v_i) = \{a_i\}$, for $1 \leq i \leq n$. In what follows we call such a graph a **linear graph**.

Definition 7. Given POs $G_1 = (V_1, E_1, \ell_1, s_1)$ and $G_2 = (V_2, E_2, \ell_2, s_2)$, and a set $PF \subseteq V_1 \times V_2$ of possible node fusions among them:

- a **compatible set of fusions** $F \subseteq PF$ is a set of node pairs that, if fused together, would combine G_1 and G_2 into a graph G that is *acyclic* (i.e., a PO);
- a **common supergraph of G_1 and G_2 defined by PF** is one of these graphs G obtained by a compatible set of fusions $F \subseteq PF$ — and is thus also a PO;
- a **Minimal Common Supergraph (MCS) of G_1 and G_2 defined by PF** is a common supergraph G defined by PF that is **minimal** (among all such supergraphs, it has minimal size).
- a **common subgraph of G_1 and G_2 defined by PF** is also a graph obtained from a compatible set of fusions $F \subseteq PF$ — specifically, the common subgraph is the subset of the supergraph G , whose nodes are all of the nodes in G resulting from fusion, and whose edges are all of the edges in G among these nodes — and because it is a subgraph of the PO G , it is also a PO;
- a **maximal common subgraph (mcs) of G_1 and G_2 defined by PF** is a common subgraph that is **maximal** (among all such subgraphs, it has maximal size).

Notice that the *MCS* and *mcs* need not be unique.

Bunke et al.^{5,6,7} have explored similar formulations for *MCS* and *mcs*. One formulation⁵ allows only identity fusions, the node+edge graph size function. Another^{6,7} permits substitution fusions and a class of size functions linear in the numbers of nodes and edges affected by the various kinds of edit operations; it defines a *graph edit distance* in terms of these size measures and shows how it is naturally related to *MCS* size.

In this paper we will emphasize the node+edge graph size measure since it is natural and it reflects graph structure. It also seems particularly useful in large-grain alignment of *segments* or *features* of sequences, and this alignment can then be refined to smaller-grain levels using other methods.

Pairwise Partial Order Alignment as a Supergraph Problem—Aligning Alignments Revisited 7

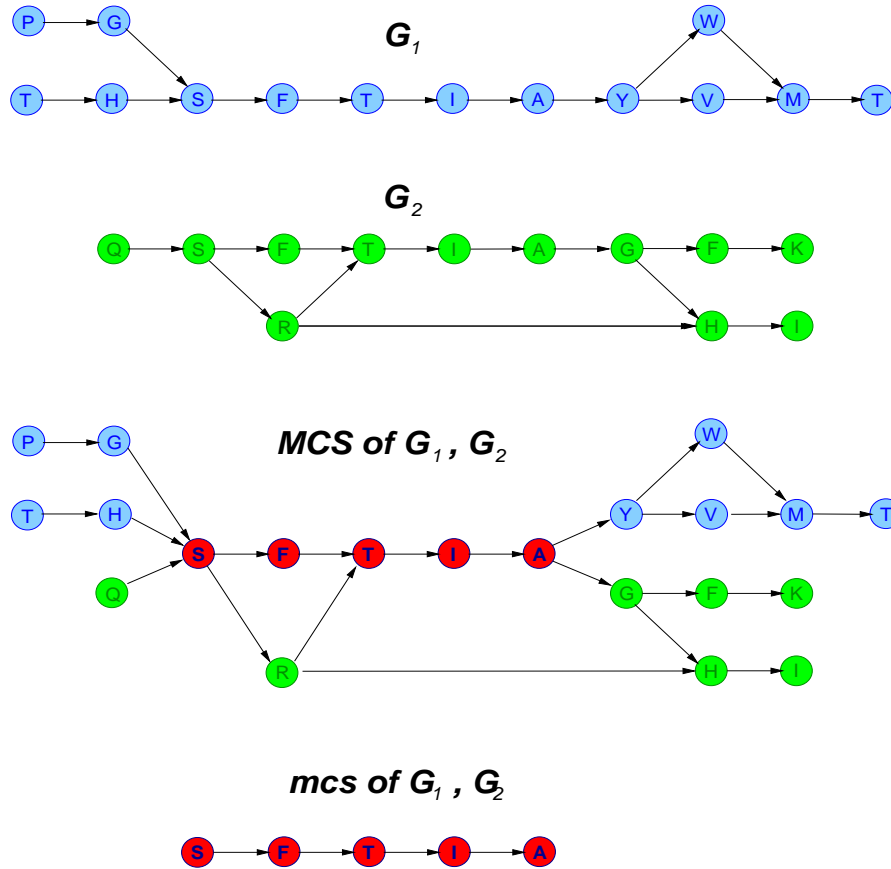


Fig. 6. A Minimal Common Supergraph (MCS) and maximal common subgraph (mcs) for two POs G_1 and G_2 , assuming the node+edge size of a graph is used. The mcs is the common subgraph induced by the nodes with labels S, F, T, I, A, and is unique for these graphs (there is no larger common subgraph). By fusing the corresponding pairs of nodes in G_1 and G_2 with these labels, we obtain a larger graph, containing both G_1 and G_2 . This larger graph is exactly a MCS of G_1 and G_2 . Thus the MCS is something like the smallest possible ‘union’ of G_1 and G_2 , and relative to this the mcs is their ‘intersection’.

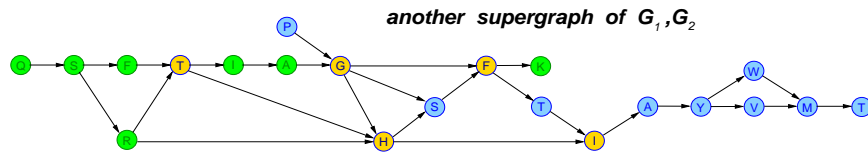


Fig. 7. Another supergraph for the two directed acyclic graphs G_1 and G_2 in Figure 6. It is not minimal if the node+edge size of a graph is used, because it has more edges than — although the same number of nodes as — the supergraph MCS of G_1 and G_2 in Figure 6. Both of these supergraphs are results of fusing five pairs of nodes between G_1 and G_2 , as is explained in Figure 8.

Definition 8. The Pairwise PO Alignment Problem:

Given POs G_1 and G_2 , and a set $PF \subseteq V_1 \times V_2$ of possible node fusions among them, find a MCS G for G_1 and G_2 using only fusions in PF .

Example 1. Each partial order alignment in Figure 1 is also a PO. Also, if the size of a graph is proportional to its number of nodes, so that minimality of a graph means it has as few nodes as possible, the fourth line of Figure 1 shows the unique solution of the Multiple PO Alignment problem for the sequences PKMIVRPQKNETV and THKMLVRNETIM.

Example 2. Figure 6 shows a MCS and a mcs for two graphs G_1 and G_2 , assuming node+edge graph size. With this size function, the MCS or mcs for the two graphs shown are unique. Figure 7 shows another graph that is obtainable by fusing a different set of five nodes in the fusion graph in Figure 8. Although it has the same number of nodes as that in Figure 6, it has more edges, and thus is not minimal.

3. Algorithms for Pairwise Partial Order Alignment

The algorithms considered in this section take two POs G_1 and G_2 as input, and find a MCS. Efficient algorithms for Partial Order Alignment exist for some special cases, such as where G_1 is a linear graph and G_2 is a PO. Also, some results are developed here for the general case.

3.1. Alignment of a Partial Order with a Sequence**Theorem 1.** (Lee, Grasso & Sharlow²¹)

In the special case where G_1 is a PO and G_2 is a linear graph (sequence), a MCS can be found in polynomial time by dynamic programming.

Specifically, if v is a node in the PO G_1 , and m is the index of a node in sequence G_2 , we obtain a dynamic programming recurrence

$$S(v, m) = \max_{(u \rightarrow v) \in G_1} \begin{cases} S(v, m-1) + \Delta(v) \\ S(u, m-1) + s(v, m) \\ S(u, m) + \Delta(m) \end{cases}$$

where u ranges over all direct predecessors of v in the partial order, $s(v, m)$ is the substitution score for aligning node v of G_1 with letter m of G_2 (fused node size), and Δ specifies a gap penalty (non-fused node size). The difference from standard sequence alignment here is that there can be multiple predecessors u for any given node v , whereas in standard alignment there can be only one. When the partial ordering G_1 is sparse, which is typical in practice, the complexity of this alignment is not much greater than that of pairwise sequence alignment²¹.

An implementation of this approach has been available (now for several years) at www.bioinformatics.ucla.edu/poa. It takes a set of sequences as input, and iteratively integrates them into a PO with dynamic programming²¹. The implementation has been used heavily in a high-throughput production environment^{18,28}.

3.2. Alignment of a Partial Order with a Partial Order

Recently Grasso¹⁴ has developed a heuristic dynamic programming algorithm for aligning two POs. The algorithm works by finding an optimal linear *mcs* for the two input POs G_1 and G_2 , directly generalizing on the algorithm just discussed for aligning a Sequence with a Partial Order. Specifically, if v and v' are nodes in G_1 and G_2 , respectively, it evaluates the dynamic programming recurrence

$$S(v, v') = \max_{\substack{(u \rightarrow v) \in G_1 \\ (u' \rightarrow v') \in G_2}} \begin{cases} S(v, u') + \Delta(v) \\ S(u, u') + s(v, v') \\ S(u, v') + \Delta(v') \end{cases}$$

where u ranges over all direct predecessors of v in G_1 , u' ranges over all direct predecessors of v' in G_2 , and $s(v, v')$ is the substitution score for aligning v of G_1 with v' of G_2 . Since only linear fusions of G_1 and G_2 are considered, the algorithm is very efficient. In a series of experiments it exhibited subquadratic time complexity, significantly outperforming CLUSTALW¹⁷ in both specific timings and in asymptotic timing behavior. It also produced superior alignments for problems generated by ROSE³¹, and for BALiBASE benchmarks³².

3.3. Computational intractability of Pairwise Partial Order Alignment

Being able to find a *MCS* for general POs G_1 and G_2 will allow us to *align alignments*. Unfortunately, in its general form this Partial Order Alignment problem is intractable:

Theorem 2. *With node+edge graph size, the pairwise Partial Order Alignment Problem is NP-complete.*

Proof. We actually show that a decision problem variant of the PO Alignment Problem is NP-complete: given a PO and a set of possible fusions, and an integer $b > 0$, does the PO have a *MCS* of size b or less, using node+edge graph size? (The corresponding optimization problem, of finding the smallest possible value of b , can be reduced to a kind of binary search using this decision problem.)

The approach here for demonstrating the hardness of this problem is to show that it contains a hard supergraph (graph subsumption) problem as a special case; deciding whether one graph is a supergraph of another is NP-complete for many kinds of graphs. The *MCS* of two POs should be one of the POs precisely when it is a supergraph of the other.

Specifically, the CLIQUE problem (a well-known NP-complete problem¹³) can be reduced to the *MCS* problem. The CLIQUE problem takes as input an undirected graph G and integer k , and asks whether G has a clique (completely-connected subgraph) of size k .

Intuitively this reduction can work by reducing the CLIQUE problem to a directed variant of the *MCS* problem. Assuming that G is an undirected graph, we convert G to a directed graph G_1 by choosing an arbitrary numbering $1, \dots, n$ of its nodes, and then orienting each of G 's edges to point from its lower-numbered node to its higher-numbered node. We also create a graph G_2 — a directed version of a clique — that has nodes numbered $1, \dots, k$, and has a directed edge from node i to node j iff $1 \leq i < j \leq k$. (We can

assume without loss of generality that $n > k$, and that G contains no isolated nodes.) Both G_1 and G_2 are directed acyclic graphs. Thus G has a clique of size k if and only if G_1 is a *MCS* of G_1 and G_2 , or equivalently, every *MCS* of G_1 and G_2 is no larger than G_1 .

Implicit in this reduction outline is the assumption that every node in both G_1 and G_2 has the same node label (e.g., letter). This is a degenerate kind of POA problem that does not arise in practice. Generally, we say that a PO is **degenerate** if it contains distinct nodes x and y that are **indistinguishable** — they have the same label, and they have the same paths: any sequence of labels on a path to (resp. from) one is also a sequence of labels on a path to (resp. from) the other. So this reduction outline is deficient in that it does not specifically show that nondegenerate POA is a hard problem.

It is possible to avoid degeneracy concerns, and reduce a CLIQUE problem to a nondegenerate POA problem, by replacing every node in G_1 and G_2 with a copy of a nondegenerate PO (one that could arise in practice), and then including a few additional edges. We do this as follows.

First, let P be a large nondegenerate PO with m source nodes and m sink nodes, where $m > (n^2 + k^2)$. (That is, in P there is a path from every source node to every other node, and to every sink node from every other node.) We then create G'_1 (resp. G'_2) by replacing every node in G_1 (resp. G_2) with a copy of P , and replacing every edge $u \rightarrow v$ in G_1 (resp. G_2) with m edges connecting the i -th sink of u 's copy of P to the i -th source of v 's copy of P ($1 \leq i \leq m$). It is still true that G has a clique of size k , and every *MCS* of G_1 and G_2 is no larger than G_1 , if and only if some *MCS* of G'_1 and G'_2 has the same number of nodes as G_1 . The point of the large value for m is that it allows us to add many additional ‘nondegeneracy-guaranteeing edges’ to G'_1 and to G'_2 without disturbing this basic structure.

So far, the graphs G'_1 and G'_2 still can be degenerate, since they are homomorphic images of G_1 and G_2 , and G_1 and G_2 can have indistinguishable nodes. This degeneracy can be avoided by adding more edges. Viewing n_u and n_v as fixed distinct integer indices for nodes u and v (ranging over the n nodes in G_1 and k nodes in G_2), for every edge $u \rightarrow v$ in G_1 (resp. G_2), we add one more edge from sink node n_u of u 's copy of P to source node n_v of v 's copy of P . There are fewer than $(n^2 + k^2)$ such edges. Adding these edges destroys any degeneracy: if nodes x and y have isomorphic predecessors and successors in G'_1 (resp. G'_2), then they cannot both be in one copy of P (since P is nondegenerate), but if they are in different copies of P , they cannot be indistinguishable, since they are not isolated (G contains no isolated nodes) and since sources and sinks of these different copies of P have been affected in non-isomorphic ways by the newly added edges.

Furthermore, since there are fewer than $(n^2 + k^2)$ such edges, adding them to G'_1 and G'_2 does not change the property that G_2 is not a k -clique in G_1 if and only if any *MCS* of G'_1 and G'_2 has at least $(n^2 + k^2)$ edges more than G_1 . (If G_2 is a k -clique in G_1 , then the *MCS* of G'_1 and G'_2 is G_1 along with the edges that were added to avoid degeneracy. If G_2 is not a k -clique in G_1 , then the *MCS* must include at least $(n^2 + k^2)$ edges more than G_1 .) So being able to bound the size of a *MCS* is sufficient to solve the CLIQUE problem.

Finally: the problem is NP-complete even when there are only two possible node labels. This construction does not impose requirements on the number of labels (letters) used. In

fact the number L of labels can be reduced to 2 by replacing each node in G'_1 and G'_2 with a sequence (linear graph) of $\lceil \log_2(L) \rceil$ nodes by using binary encodings. \square

This is consistent with other negative results obtained recently for multiple sequence alignment and supergraph problems^{3,20,35}.

For two graphs, when the node+edge graph size function is used, the *MCS* problem can be treated as equivalent to the *mcs* problem⁷. The *mcs* problem was long ago shown to be NP-complete for general graphs¹³, using a similar transformation from CLIQUE. The close connections between POA and CLIQUE will now be explored.

3.4. Fusion Graphs: graphical representations of compatible sets of fusions

Finding a *mcs* and finding a *MCS* can be computationally equivalent, in the sense that finding an instance of one easily yields an instance of the other. Furthermore, under certain assumptions on the graph size measure, the *MCS* and *mcs* of two graphs can be directly related⁷. For example, when the size of a graph is $|V|$, any set of fusions F yielding a *mcs* also yields a *MCS*, since both depend only on the size $|V'|$ of the set of fused nodes. This also is true for *node+edge graph size*, since again the number of nodes is of primary importance.

Many algorithms for finding a *mcs* are now available^{1,25,27,30}. Some of these rest on clique finding algorithms^{1,7,5,8}. Our idea is to treat the *mcs* problem for G_1 and G_2 as a maximal clique problem on a corresponding *compatibility graph* of G_1 and G_2 , which is related to their fusion graph.

Definition 9. The **fusion graph** $FG = (PF, CF)$ of two POs G_1 and G_2 is a directed graph. The set of nodes $PF \subseteq V_1 \times V_2$ defines all possible fusions (u, v) of pairs of nodes $u \in V_1, v \in V_2$.

The edges CF of the fusion graph are then defined as follows. For distinct (u, v) and (u', v') in PF , there is a directed edge from (u, v) to (u', v') in CF if joining the POs G_1 and G_2 by fusing u with v and fusing u' with v' yields a graph in which there is a path from u to u' (equivalently, from v to v').

Thus it is possible for there to be an edge from (u, v) to (u', v') and an edge from (u', v') to (u, v) . Two such nodes are called **pairwise incompatible fusions**; otherwise they are **pairwise compatible**.

Example 3. Figure 8 shows the fusion graph for the graphs G_1 and G_2 that appeared earlier in Figure 6. The fusion graph has 11 nodes, which is the total number of times a node label in G_1 matches a node label in G_2 .

The special case of pairwise sequence alignment is helpful for gaining intuition about fusion graphs. For two input sequences, in order to solve the PO Alignment Problem we must find a directed acyclic graph that contains each of the sequences and is also minimal, i.e., has as few nodes as possible. Finding such a graph is equivalent to finding a **LCS (Longest Common Subsequence)**¹⁶ of the two input sequences. Figure 9 shows the fusion graph for the two sequences PKMIVRPQKNETV and THKMLVRNETIM, viewing each

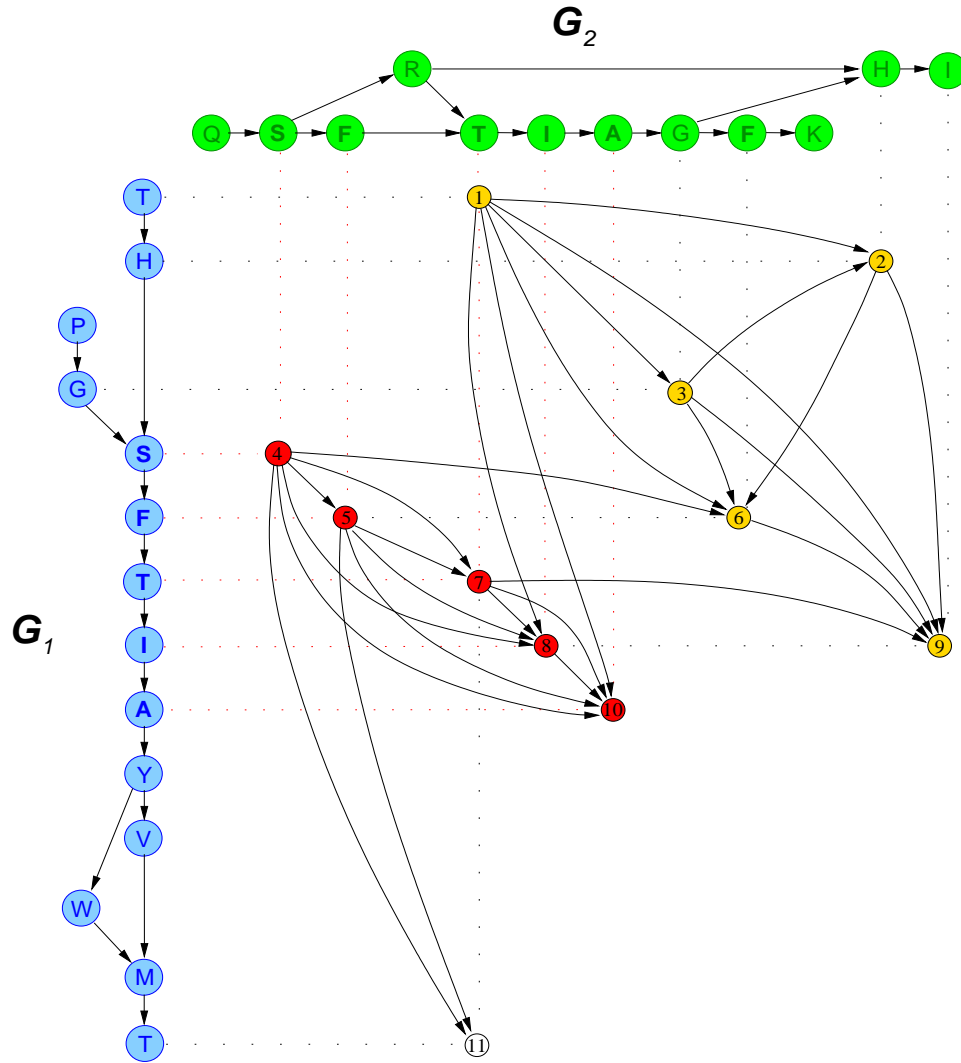


Fig. 8. The fusion graph for the graphs shown in Figure 6. For clarity, edges between pairwise incompatible fusion nodes have been omitted. Nodes in the fusion graph correspond to node pairs (u, v) (u in G_1 , v in G_2) having identical labels. Highlighted nodes identify the maximal sets of compatible fusions. As an example, there is no edge between nodes 1 and 4 of this graph, because they represent pairwise incompatible fusions: node 1 corresponds to fusing the T at the start of G_1 with the T in the middle of G_2 , while node 4 corresponds to fusing the S near the middle of G_1 with the S near the start of G_2 . Performing both of these fusions on G_1 and G_2 would yield a cyclic graph. By contrast, any set of mutually-compatible fusions will yield a supergraph of G_1 and G_2 that is a PO. Specifically, performing the highlighted sets of fusions $\{4, 5, 7, 8, 10\}$ will yield the supergraph shown in Figure 6, which is a MCS, and performing the fusions $\{1, 2, 3, 6, 9\}$ will yield the supergraph shown in Figure 7, which is not minimal because its number of edges is not minimal.

sequence as a graph. A POA is obtained from these two by making fusions corresponding to compatible nodes in the fusion graph.

Theorem 3. *For POs G_1 and G_2 , the set of Minimal Common Supergraphs is in 1-to-1 correspondence with the set of maximal directed acyclic subgraphs of their fusion graph.*

Proof. The union of G_1 and G_2 is a PO — a directed acyclic graph. Any set F of valid pairwise node fusions (subset $F \subseteq PF$ of nodes in the fusion graph) will yield a common supergraph of the two. We claim that this supergraph is a PO (directed acyclic graph) if and only if the nodes in F define a directed acyclic subgraph of FG .

Equivalently: *performing a set of fusions F on G_1 and G_2 yields a cycle if and only if F defines a corresponding cycle in the fusion graph FG .* We prove this by induction on the number n of nodes in F . The basis for $n = 1$ is trivial, since no single fusion can yield a cycle. For the induction step, assume $F = \{(u_1, v_1), \dots, (u_n, v_n)\}$ with $n > 1$.

(\Rightarrow) Suppose the fusions of F yield a cycle in the union of G_1 and G_2 . This cycle must include fused nodes in some order, since without the fused nodes there is no cycle. We can assume no proper subset of F has this property, since otherwise the rest follows by induction. Thus we can assume the cycle is of the form

$$w_1 \xrightarrow{*} w_2 \xrightarrow{*} \dots \xrightarrow{*} w_n \xrightarrow{*} w_1$$

where each node w_i is the fusion of node u_i in G_1 with node v_i in G_2 , and each arrow denotes either a path that is in G_1 or a path that is in G_2 . The fusion graph definition requires that each path between w_i and w_{i+1} have a corresponding edge $((u_i, v_i) \rightarrow (u_{i+1}, v_{i+1}))$ in FG . So F defines a cyclic subgraph of the fusion graph FG .

(\Leftarrow) Suppose that F defines a cycle in FG . We can assume no proper subset of F does also, since otherwise the rest follows by induction. Thus we can assume (by renumbering if necessary) that FG contains the edges

$$((u_1, v_1) \rightarrow (u_2, v_2)), \dots, ((u_{n-1}, v_{n-1}) \rightarrow (u_n, v_n)), ((u_n, v_n) \rightarrow (u_1, v_1)).$$

By the fusion graph definition, the i -th edge corresponds to a path in the union of G_1 and G_2 from either u_i or v_i to either u_{i+1} or v_{i+1} . With the fusions of F , these paths define a cycle in the union of G_1 and G_2 . \square

The fusion graph is primarily a bookkeeping construction that allows us to find compatible sets of fusions. In fact the result above can be generalized for multiple graphs²⁴.

3.5. Implementing Pairwise Partial Order Alignment

Theorem 3 shows that implementation of POA requires finding good sets of node fusions. Specifically, the fusions must be *compatible* — i.e., performing them does not create a cycle, and destroy the partial ordering property. (Some time ago by Morgenstern et al.²⁹ mentioned a similar idea of using partial order as a consistency check for proposed MSAs.) Beyond this, the fusions should be as extensive (size-reducing) as possible, so as to yield a minimal supergraph.

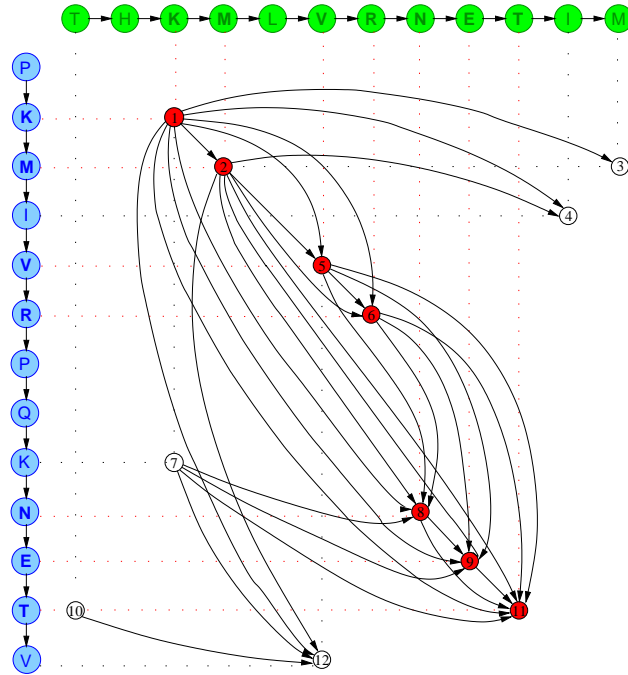


Fig. 9. The fusion graph for the graphs corresponding to sequences $PKMIVRPQKNETV$ and $THKMLVRNETM$. For clarity, the edges between incompatible fusion nodes have been omitted. Furthermore, the nodes PF in this graph correspond to identity fusions (fusion of nodes with identical labels). The highlighted nodes identify the unique maximal compatible set of fusions among these (yielding a minimal supergraph of the sequences). $KMVRNET$ is the unique LCS (Longest Common Subsequence) of these two sequences. In the special case of two sequences, the fusion graph 'grid' illustrated in this figure (in which the coordinates are pairs (u, v) of indexes into the two sequences) is directly related to the usual dynamic programming alignment matrix. Two nodes in the fusion graph are compatible if and only if one is diagonally above and to the left of the other. Any set of fusions is thus a sequence alignment 'diagonal', and a maximal set of fusions corresponds both to a LCS of the sequences and to a mcs of their graphs. The resulting POA is minimal when as many nodes are fused as possible, which is achieved by performing the (in this case unique) maximal set of fusions of this fusion graph. Notice the maximal set corresponds precisely to the subsequence $KMVRNET$, which was shown in the fourth line of Figure 1 to be the LCS for these sequences.

If a set of fusions F is compatible, then all of its elements must be pairwise compatible. Construct a graph FG' that has the same nodes PF as the fusion graph FG , but has (undirected) edges between pairwise compatible fusions. If F is a compatible set of fusions, then, F will be a clique of FG' — a set of nodes that is completely connected in FG' . So, we can implement POA by finding large cliques F of FG' , checking that they are also compatible subsets of FG , and if they are, then determining the size of the supergraph that results by performing all the fusions in F . Any smallest such supergraph is a POA. If the size of a graph is a monotonic function of its number of nodes, or if performing graph fusions cannot increase the size of a graph, then search can be restricted to *maximal* cliques.

There are many algorithms for finding maximal cliques⁹. A popular choice has been the *Bron-Kerbosch* clique-finding algorithm⁴, a clique enumeration method that includes a branch-and-bound technique for avoiding rediscovery of cliques, and heuristics to improve search performance. It has the virtue that it can be easily modified to return ‘all reasonable cliques’. Furthermore its greedy strategy often succeeds in finding large cliques rapidly, and as a result it often can generate respectable suboptimal solutions quickly, even when the search space is too large to be exhaustively searched. The Bron-Kerbosch algorithm has been used to find common subgraphs and supergraphs in bioinformatics applications in the past, notably in comparing molecular structure. For example, for over a decade it has been used in production protein structure alignment^{15,34}, and protein threading²⁶, which involves the alignment of protein sequences with protein structures. VAST, the NCBI Vector Alignment Search Tool³³ adopts this approach, and it is used in NCBI’s MMDB¹⁰ to relate protein domains in PDB.

In order to implement POA, we implemented an extension of the Bron-Kerbosch algorithm to find maximal fusion sets. Our program consists of about 2500 lines of C, of which 200 lines is the Bron-Kerbosch algorithm, 200 lines is acyclicity checking, 650 lines is Minimal Common Supergraph construction, and most of the remainder is concerned with building, manipulating, and reading and printing of graphs. The POA output from the program is rendered as a graph, using a layout obtained from the *dot* program in the *graphviz* package¹² available from AT&T Research; some layouts are shown in Figures 10 and 11.

The graph formulation of POA presented here gives a new way to attack the difficult problem of *aligning alignments*¹⁹. Examples of alignments of alignments produced by the program appear in Figure 10. This problem has high intrinsic complexity, and POA does not change this. However POA admits heuristics that may work well in practice, such as the recent work of Grasso¹⁴ mentioned above. Also, the shift from tabular layouts to partial order graphs does change the problem in ways that might lead to new approaches.

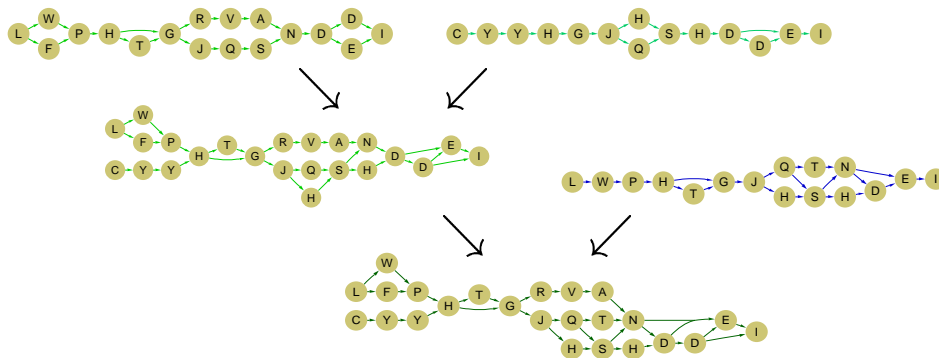


Fig. 10. *Progressive alignment of alignments*. Iterative pairwise alignment can be used to align alignments; the two operations shown here obtained Minimal Common Supergraphs of their operands (assuming node+edge graph size, and identity fusions) using the clique-finding MCS implementation. While progressive alignment algorithms often generate their plan for merging alignments by using a similarity measure among sequences, here it is important to choose pairs of alignments to align so as to minimize the size of their fusion graphs.

Hs#S672182	a.gttcctgctgctgttggactgatgtacttg.ttt.....
Hs#S337687	aagttcctgctgctgttggactgatgtacttggttgnaggcaa
Hs#S629177	a.gttcctgctgctgttggactgatgtacttg.tttgt.naggcaa
Hs#S1794113	a.gttcctgctgctgttggactgatgtacttg.tttgtg.aggcaa
Hs#S813765	a.g.tcctgc.gcgtttgc.ggacggatgtacttg..ttgtg.aggcaa
Hs#S663801	a.gttcctgctgctgttggacttatgtacttg.tttgtg.aggcaa
Hs#S4698	a.gttcctgctgctgttggactgatgtacttg.tttgtg.cggcaa
Hs#S1988018	a.gttcctgctgcttttggactgatgtacttg.attgtg.aggcaa
Hs#S196113	a.gttntgntgnttggactgatgtacttg.tttgtg.aggcaa



Fig. 11. A supergraph for the indicated Hs.100194 sequence segments from Figure 2. The result is comparable to Figure 3. Although layouts of this kind lose information about individual sequences, this information can be retained with schemes that identify individual paths, such as in the final POA shown here. The two graphs shown are actually both renderings of one graph using different dot program parameters. If the sequences are numbered from 0 to 8, the result was produced by the arbitrarily-chosen pairwise alignments $((0 (1 2)) 3) (((4 5) 6) 7) 8)$. For perspective on complexity, consider the problem of aligning the first two sequences. If each nucleotide is represented as a graph node, the fusion graph has 510 vertices and 198167 edges. The large number of possible identity fusions results from the small nucleotide alphabet. Although this brute-force approach to fusion is impractical, the Bron-Kerbosch algorithm is often able to find large cliques quickly. However, an improvement limits the search dramatically: by representing subsequences of 3 nucleotides as single graph nodes, the resulting fusion graph shrinks to 84 vertices and 4487 edges, with a unique maximal clique of 43 fusions. This clique is found by Bron-Kerbosch in less than a second on a 750 MHz SUNW UltraSPARC-III CPU, and the resulting alignment is optimal. Ultimately the POA for all sequences was produced in 23 seconds. With greater ingenuity the search can be reduced further.

As another application, we had the program iteratively find pairwise MCS for some Hs.100194 sequence fragments (like Figure 3 for the sequences in Figure 2). The resulting POA is shown in Figure 11, retaining the paths that correspond to each sequence.

The more that searches can be limited to fusion of *features* like blocks or segments, rather than fusion of individual residues, the larger the problems that can be handled. For example, Figure 5 presents a POA for four protein sequences, showing 4 ‘hits’ (significant aligned segments) among them. If the sequences are then represented in terms of hit segments, the resulting POA problem is easily solved with clique-based approaches.

4. Conclusion

Partial Order Alignment (POA)^{14,18,21,23,28} presents an alternative to conventional sequence alignment. This paper is an attempt to characterize POA in graph-theoretic terms. From this perspective pairwise POA amounts to the problem of finding a minimal supergraph for the two input graphs. Although this problem is NP-complete, this formulation is helpful conceptually and can lead to new algorithms. For example, this formulation suggests an implementation based on clique finding, and for large-grain alignment problems (alignment of hits, HSPs, blocks, domains, ...) these clique methods can be practical. This

formulation also gives new ways to approach the difficult problem of *aligning alignments*.

A companion paper²⁴ analyzes analogues of results in this paper for multiple alignment. Pairwise and multiple alignment differ in interesting ways. First, by forming the union of the input graphs, with an appropriate definition of *PF* the multiple alignment problem can be restricted to take a single graph as input. Second, pairwise alignment fusions always combine a pair of nodes from the two input graphs, where multiple alignment fusions can combine arbitrarily many nodes from the single input graph.

This paper has concentrated on presenting the POA concept in terms of graphs, leaving open many specific questions about implementation. The aim has been to develop a foundation for further exploration and development. An implementation and other information about POA is available at www.bioinformatics.ucla.edu/poa.

Acknowledgement

We gratefully acknowledge the many contributions of Catie Grasso to this work. We are also grateful to Mike Quist for comments that have improved this paper. Discussions with Rachel Kolodny were very helpful in addressing the computational complexity of POA. A conversation with Pavel Pevzner was also important in clarifying points about the exposition and about implementation. This research was supported by NSF grant IIS 0082964, NIH grant MH065166, and by the *Foundation for Prevention of Total Order*.

References

1. H.G. Barrow, R.M. Burstall, “Subgraph isomorphism, matching relational structures and maximal cliques”, *Information Processing Letters* 4:4, 83–84, 1976.
2. NCBI BLAST, home page: www.ncbi.nlm.nih.gov/BLAST/
3. P. Bonizzoni, G. Della Vedova, “The complexity of multiple sequence alignment with SP-score that is a metric”, *Theoretical Computer Science* 259:1–2, 63–79, 2001. citeseer.nj.nec.com/506041.html
4. C. Bron, J. Kerbosch, “Algorithm 457: Finding all cliques of an undirected graph”, *Comm. ACM* 16, 575–577, 1973.
5. H. Bunke, K. Shearer, “A graph distance metric based on the maximal common subgraph”, *Pattern Recognition Letters* 19:3–4, 255–259, 1998. www.iam.unibe.ch/~fki/publications/papersOnGraphMatching/a-graph-distance-metric.ps.gz
6. H. Bunke, “Error correcting graph matching: On the influence of the underlying cost function”, *IEEE Trans. Pattern Analysis and Machine Intelligence*, 21:9, 917–922, 1999. www.iam.unibe.ch/~fki/publications/papersOnGraphMatching/errorCorrecting.ps.gz
7. H. Bunke, X. Jiang, A. Kandel, “On the Minimum Common Supergraph of Two Graphs”, *Computing* 65, 13–25, 2000. www.iam.unibe.ch/~fki/publications/papersOnGraphMatching/CommonSupergraphV2.ps.gz
8. H. Bunke et al., “A comparison of algorithms for maximum common subgraph on randomly connected graphs”, *Proc. IAPR Workshop on Structural and Syntactic Pattern Recognition*, 2002. www.iam.unibe.ch/~fki/publications/papersOnGraphMatching/SSPR2002.ps.gz
9. I. M. Bomze, M. Budinich, P. M. Pardalos, M. Pelillo. “The maximum clique problem”, in D.-Z. Du and P. M. Pardalos, eds., *Handbook of Combinatorial Optimization*, volume 4, Kluwer Academic Publishers, Boston, MA, 1999. citeseer.nj.nec.com/bomze99maximum.html
10. J. Chen et al., “MMDB: Entrez’s 3D-structure database”, *Nucleic Acids Res.* 31:1, 474–477, 2003. www.ncbi.nlm.nih.gov/Structure/MMDB/mmdb.shtml

11. R. Durbin et al., *Biological sequence analysis*, Cambridge University Press, 1998.
12. E.R. Gansner, E. Koutsofios, S.C. North, and K.P. Vo, "A technique for drawing directed graphs," *IEEE Trans. Soft. Eng.* 19, 214–230, 1993. www.research.att.com/sw/tools/graphviz
13. M.R. Garey, D.S. Johnson, *Computers and Intractability: a Guide to the Theory of NP Completeness*, W.F. Freeman and Sons, 1979.
14. C. Grasso, C. Lee, "Applying Partial Order Alignment to Progressive Multiple Sequence Alignment of Proteins", 2003. Submitted for publication.
15. H.M. Grindley et al., "Identification of tertiary structure resemblance in proteins using a maximal common subgraph isomorphism algorithm", *J. Mol. Biol.* 229, 707–721, 1993.
16. D. Gusfield, *Algorithms on strings, trees, and sequences: computer science and computational biology*. Cambridge University Press, 1997.
17. D. Higgins et al., "CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice." *Nucleic Acids Res.* 22:4673–4680, 1994. www.ebi.ac.uk/clustalw/
18. K. Irizarry, et al., C. Lee, "Genome-wide analysis of single-nucleotide polymorphisms in human expressed sequences," *Nature Genetics* 26, 233–236, October 2000.
19. J. D. Kececioğlu and W. Zhang, "Aligning alignments", in M. Farach, ed., *Proc. CPM (Combinatorial Pattern Matching) '98*, LNCS # 1448, 189–208, Springer-Verlag, 1998. citeseer.nj.nec.com/kececioğlu98aligning.html
20. P. Kilpeläinen and H. Mannila, "Ordered and unordered tree inclusion", *SIAM Journal on Computing*, 24:2, 340–356, April 1995.
21. C. Lee, C. Grasso, and M. Sharlow, "Multiple sequence alignment using partial order graphs", *Bioinformatics* 18, 452–464, 2002. www.bioinformatics.ucla.edu/poa/
22. T. Lassman, E. Sonnhammer, "Quality assessment of multiple alignment programs", *FEBS Letters* 529, 126–130, 2002.
23. C. Lee, "Generating Consensus Sequences from Partial Order Multiple Sequence Alignment Graphs," *Bioinformatics*, 2003.
24. C. Lee, D.S. Parker, "Multiple Partial Order Alignment as a Graph Problem", submitted for publication, September 2003.
25. G. Levi, "A note on the derivation of maximal common subgraphs of two directed or undirected graphs", *Calcolo* 9, 341–354, 1972.
26. T. Madej et al., "Threading a database of protein cores", *Proteins* 23, 356–369, 1995.
27. J.J. McGregor, "Backtrack search algorithms and the maximal common subgraph problem", *Software – Practice & Experience* 12, 23–34, 1982.
28. B. Modrek, A. Resch, C. Grasso, C. Lee, "Genome-wide detection of alternative splicing expressed sequences of human genes," *Nucleic Acids Research* 29:13, 2850–2859, October 2001.
29. B. Morgenstern, A. Dress, T. Werner, "Multiple DNA and protein sequence alignment based on segment-to-segment comparison", *Proc Natl Acad Sci U S A* 93, 12098–12103, 1996.
30. K. Shearer, S. Venkatesh, H. Bunke, "An efficient least common subgraph algorithm for video indexing", *Proc. 14th ICPR Conf.*, Brisbane, 1241–1243, 1998. www.iam.unibe.ch/~fki/publications/papersOnGraphMatching/an-efficient-least-common.ps.gz
31. J. Stoye et al., "ROSE: generating sequence families", *Bioinformatics* 14, 157–163, 1998.
32. J.D. Thompson, F. Plewniak, O. Poch, "BAliBASE: a benchmark alignment database for the evaluation of multiple alignment programs", *Bioinformatics* 15, 87–88, 1999.
33. VAST (Vector Alignment Search Tool) www.ncbi.nlm.nih.gov/Structure/VAST/vast.shtml
34. P. Willett, "Matching of chemical and biological structures using subgraph and maximal common subgraph isomorphism algorithms", in: D.G. Truhlar et al. (eds.), *Rational drug design*, NY: Springer-Verlag, 11–38, 1999.
35. A. Yamaguchi, K. Nakano, and S. Miyano, "An Approximation Algorithm for the Minimum Common Supertree Problem", *Nordic J. Computing* 4:3, 303–316, Fall 1997.