Proof Paper #2 Rough Draft

Paul Bodily

August 21, 2014

1 Background

Accurate and efficient genome assembly algorithms are essential in unlocking the solutions to challenges posed by genetic disease, genetic engineering, and even next-generation digital information storage [1]. The term *genome* describes a complete set of DNA in the cell of an organism. Within the genome, DNA is organized as a distinct set of molecules called *chromosomes*, the number of which is a unique characteristic of a species. Each chromosome is composed of a sequence of *nucleotide bases*, or simply *nucleotides*, which encode all of the functionality of a living organism. The goal of genome assembly is to ascertain the identity of this sequence and is prerequisite to making inferences about the complex mechanisms that govern life.

DNA is inherently directional, which means that defining the head of a sequence (called the $5'$ or *five prime* end) and the tail of a sequence ($3'$) is as essential as defining the nucleotide residues themselves. By convention, the *forward strand* of a sequence s^+ is written in the $5'$ to $3'$ direction (the same order in which DNA is biologically replicated, transcribed, and sequenced); however, as DNA is *double-stranded*, it is always implied (though usually not written) that for any sequence s^+, an equally viable reverse-complement sequence s^- exists whose $5'$ to $3'$ direction is opposite that of s^+. We will define moving in the $5'$ to $3'$ direction as moving *downstream* and $3'$ to $5'$ as moving *upstream* (see Fig. 1).

Predominant next-generation sequencing technologies are only capable of *sequencing* (i.e., ascertaining the sequence of nucleotide residues of) short DNA fragments called *reads*. Consequently the genome assembly process requires first randomly fragmenting several copies of the chromosomes (which in humans are on the order of hundreds of millions of nucleotides in length) into small segments to be sequenced. Then

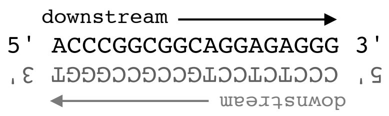


Figure 1: By convention DNA is written $5'$ to $3'$. DNA is double-stranded with $5'$ to $3'$ directionality of the reverse-complement strand being opposite that of the forward strand.

```
Read 1:     ACCCGGCGGCAGGAGAGGGGATGAAGATGGCGGACGCGAAGC
Read 2:      CCCGGCGGCAGGAGAGGGGATGAAGATGGCGGACGCGAAGCA
Read 3:       CCGGCGGCAGGAGAGGGGATGAAGATGGCGGACGCGAAGCAG
Read 4:        CGGCGGCAGGAGAGGGGATGAAGATGGCGGACGCGAAGCAGA
Read 5:         GGCGGCAGGAGAGGGGATGAAGATGGCGGACGCGAAGCAGAA
Read 6:          GCGGCAGGAGAGGGGATGAAGATGGCGGACGCGAAGCAGAAG
Read 7:           CGGCAGGAGAGGGGATGAAGATGGCGGACGCGAAGCAGAAGC
...

Consensus:  ACCCGGCGGCAGGAGAGGGGATGAAGATGGCGGACGCGAAGCAGAAGC...
```

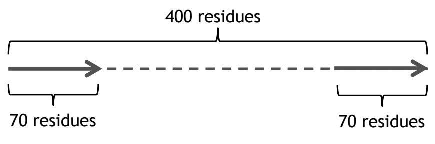
Figure 2: The contigging phase overlaps short DNA reads to determine the consensus sequence of the larger derivative molecule.

confident overlaps are found between the sequenced *reads* to recover the chromosomal sequence. This process is known as *contigging* (see Fig. 2).

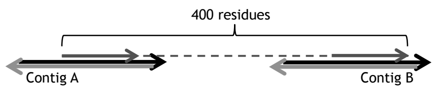
In reality, insufficient molecular sampling and repetitive regions in the DNA prevent full chromosomal reconstruction. It is thus commonplace for assembly algorithms to produce a large set of partially-reconstructed chromosomes termed *contigs*. Contigs must then be oriented and positioned relative to one another in order to reconstruct full chromosomes through *scaffolding*.

Scaffolding uses relatively long fragments of an approximately known length whose ends are sequenced (called *paired reads*) to infer positional and orientational relationships between contigs (see Fig. 3). A paired read may match to either the forward sequence or the reverse-complement sequence and thus there are four possible ways in which two adjacent contigs may be oriented relative to one another (see Fig. 3c). A *scaffolding* of two contigs is defined as the relative positioning and orientation of contigs as indicated by paired read evidence. A particular scaffolding of two contigs may be supported by multiple paired reads, which increases the confidence in the scaffolding.

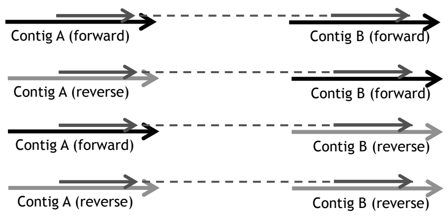
The problem of scaffolding contigs is commonly modeled as a graph (i.e., a *scaffold graph*) where vertices are contigs and edges indicate scaffoldings of contigs, weighted by the amount of supportive evidence for the scaffolding. In the contigging phase, reads from repetitive regions in the genome will overlap and combine into a single contig. Contigs representing repetitive sequence naturally suggest varied scaffoldings that create conflicts in the scaffold graph. Sequencing errors create additional complexity, suggesting contig positionings and/or orientations that are inconsistent with other scaffolding evidence. In the graph that results from these complications, the goal becomes finding a maximal path that incorporates each contig sequence once. The problem we are concerned with, the contig orientation problem, is a preparatory step to finding such a path. In the reconstruction, it is often assumed that each sequence may be included in the path only once (i.e., loops are disallowed) and that therefore each contig will be assigned a single orientation in the reconstruction. The *contig orientation problem* describes the preliminary challenge of assigning contig orientations so as to minimize conflicting orientation evidence. More specifically, the goal is to remove the minimum number of edges from the scaffold graph so that the remaining subgraph



(a) Paired Reads



(b) Scaffolding



(c) Orientation of Contigs in Scaffolds

Figure 3: a) In general, paired reads are reads that derive from a long DNA fragment whose ends have been sequenced (arrow indicates the $5'$ to $3'$ directionality). b) Paired reads are used to infer the relative position and orientation of contigs to which the ends find matching locations. c) Although the paired reads have a fixed position and orientation, depending on whether an end aligns to a contig's forward (black) or reverse strand (grey), contig pairs may be oriented in four different ways within a scaffold.

suggests a single consistent orientation of all vertices in the graph [3].

2 Definitions

In this section we formally define the contig orientation problem as the MAX-DIR problem. Our theorem, detailed formally in section 3, states that this problem is NP-complete.

2.1 Bidirected Scaffold Graphs

A *bidirected graph* (as introduced by [2]) is formally defined as an undirected multigraph G with a set of vertices V and a set of *bidirected edges* E (recall that a multigraph allows *multiple* edges between two given nodes or equivalently an *edge weight*). A bidirected edge e is a five-tuple (v_i, o_i, v_j, o_j, w) consisting of two vertices, v_i and v_j, the weight of the edge, w, and two *endpoint orientations*, o_i and o_j, one with respect to each of its vertices. An endpoint orientation may be either *positive* or *negative*, defining e as either *positive-incident* or *negative-incident* to the corresponding endpoint.

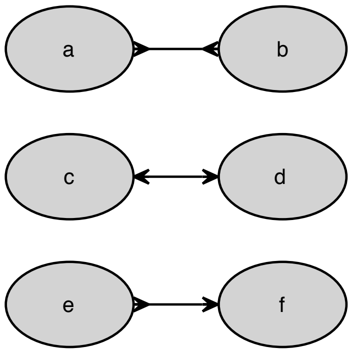


Figure 4: There are three forms of bidirected edges: *introverted*, *extraverted*, and *directed*.

In the graphical representation of a bidirected edge e, we represent positive-incidence with an arrow pointing out of the vertex and negative-incidence with an arrow pointing in to the vertex. Based on this representation, we say that e is: *introverted* if positive-incident to both endpoints; *extraverted* if negative-incident to both endpoints; and *directed* if it is positive-incident to one endpoint and negative-incident to the other (see Fig. 4). A *directed graph* is a special case of a bidirected graph in which all edges are directed edges.

A valid (v_1,v_k)-*walk* is a sequence v_1,e_1,. . .,v_{k-1},e_{k-1},v_k where e_i is an edge incident to v_i and v_{i+1} and for all $2 \leq i \leq k-1$, e_{i-1} and e_i have opposite endpoint orientations incident to v_i (see Fig. 5).

Based upon this definition, we define a *bidirected scaffold graph* for a set of contigs C and a set of weighted scaffoldings F as a bidirected graph G=(V,E) in which vertex $v_i \in V$ represents contig $c_i \in C$ and a weighted bidirected edge e=(v_i, o_i, v_j, o_j, w) represents the scaffolding $f \in F$ of contigs c_i and c_j, weighted by the number of supporting paired reads (e.g., see Figure 6b). The endpoint orientations, o_i and o_j, are determined by the relative orientation of the forward strands of c_i and c_j in f—if the forward strands of c_i and c_j are oriented in the same direction, then e is a directed edge that is positive-incident to the vertex representing the upstream contig; if the forward strands are oriented away from one another (i.e., $5'$ ends are proximal), then e is an extraverted edge; and if the forward strands are oriented towards one another (i.e., the $3'$ ends are proximal), then e is an introverted edge.

Two critical specifications must be made at this point to the bidirected scaffold graph in order that our biological constraints are maintained. First, the $5'$ to $3'$ directionality of a DNA molecule must be consistent along the entire length of the sequence. This means that introverted and extraverted edges, both of which represent internally inconsistent $5'$ to $3'$ directionality of the forward strand, violate a biological constraint. As per this definition, only directed edges are considered *valid* in the final scaffold reconstruction.

The second specification, however, is that because the biological molecule allows us to consider a contig c_i as either its forward strand c_i^+ or its reverse-complement

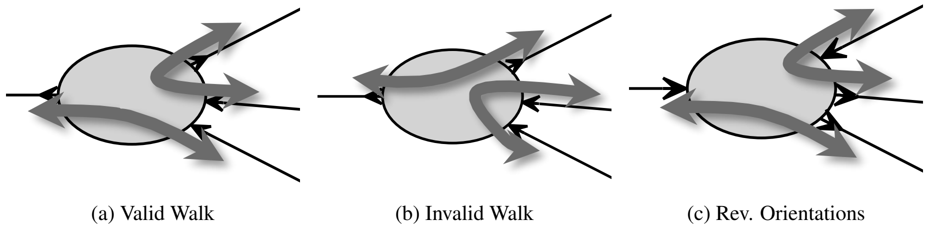


Figure 5: (a) In a bidirected scaffold graph, a valid walk enters and exits a node through oppositely-oriented arrows. (b) An invalid walk enters and exits a node through same-oriented arrows. (c) Reversing all edge-orientations adjacent to a node results in the same potential valid walks. This is biologically equivalent to selecting the reverse-complement s^- in place of the forward sequence s^+.

strand c_i^-, for any vertex $v_i \in V$ we can arbitrarily select between the *forward-orientation assignment* v_i^+ and the *reverse-orientation assignment* v_i^- in order to ensure consistency of the $5'$ to $3'$ directionality of contigs c_i and c_j in scaffolding f (the forward-orientation assignment is the vertex configuration as it is initially constructed). We will refer to this selection as the *contig-orientation assignment* of c_i or *vertex-orientation assignment* for v_i. Furthermore we refer to a contig-orientation assignment for all contigs in C (or vertices in G) as a *contig-orientation assignment* of C (or *vertex-orientation assignment* of G).

Switching between vertex-orientation assignments v_i^+ and v_i^- for vertex v_i is equivalent to flipping all endpoint orientations incident to v_i (see Fig. 5c), thus potentially rendering introverted and extraverted edges as directed edges, or vice versa. We will refer to a vertex v_i with possible vertex-orientation assignments v_i^+ and v_i^- as an *orientable vertex*. As the forward strand c_i^+ of each assembled contig c_i is arbitrarily given as input, each corresponding vertex v_i is initially assumed to be assigned the vertex-orientation v_i^+.

The goal of finding a maximal contig-orientation assignment for a graph is not equivalent to the goal of finding a maximal walk through the scaffold graph (i.e., creating linear scaffolds). There are several cases (e.g., heterozygous sequences and/or repetitive sequences on different chromosomes) in which a single connected contig-graph component can account for *multiple* walks resulting in multiple scaffolds that belong in the final reconstruction. Indeed further algorithmic elucidation of reconstructive paths is required following the resolution of the contig orientation problem. What *will* be guaranteed from resolution to the contig orientation problem is that any valid walk chosen from the subgraph of all nodes and all directed edges will have consistent internal orientation of all constituent contigs. A maximal solution to the contig orientation problem seeks to accomplish this while maximizing the weight of the directed edges.

2.2 MAX-DIR Problem

We formally state the corresponding decision problem as follows (see also Fig. 6b):

MAX-DIR $= \{(G,k) \mid G$ is a bidirected graph with orientable vertices, and there exists a vertex-orientation assignment for G yielding k directed edges$\}$

Depending on whether the preferred bias is towards including more weight or more edges, the weighted and unweighted versions of this problem (respectively) become important.

3 Theorem

3.1 Theorem Statement

MAX-DIR is NP-complete.

Proof. To prove this theorem we must demonstrate that

1. MAX-DIR $\in$ NP and
2. $\forall L \in$ NP, $L \leq_m^P$ MAX-DIR.

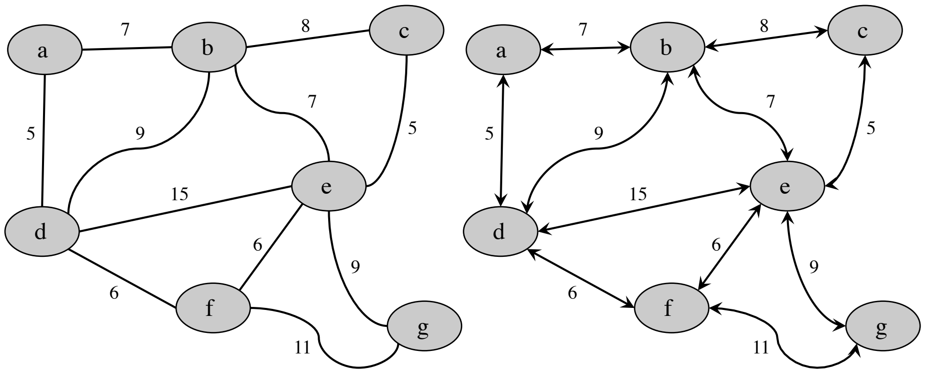
We can easily show that MAX-DIR is in NP by noting that given a vertex-orientation assignment to a bidirected graph with orientable vertices and an integer k, we can check in polynomial time whether the assignment yields k directed edges.

To prove that $\forall L \in$ NP, $L \leq_m^P$ MAX-DIR, we need merely show that some other NP-complete problem is many-one reducible in polynomial time to MAX-DIR. We will demonstrate that MAX-CUT, a well-known NP-complete problem, has such a reduction to MAX-DIR. Recall that the decision problem corresponding to the MAX-CUT problem is as follows:

$$\text{MAX-CUT} = \{(M,\, k) \mid M \text{ is a multigraph with a cut of size } k\}$$

where a *multigraph* M=(V,E) is a graph allowing multiple edges between two nodes and a *cut* in a graph is a partition of V into two distinct subsets S and T (see Figure 6a). The size of the cut is the number of edges $e \in E$ which have an endpoint in S and an endpoint in T. We will describe a polynomial-time-bounded construction that maps an instance (M,k) of MAX-CUT to some bidirected graph with orientable vertices G and positive integer k such that M has a cut of size at least k if and only if G has a vertex-orientation assignment yielding k directed edges. Let V and E be the vertex and edge sets of M and let V' and E' be the vertex and edge sets of G which we will create. The construction of G consists of the following steps (see Figure 6b):

1. Let V'= V.
2. For each edge $e \in E$ linking vertices v_i,$v_j \in V$, we create a bidirected edge e' linking v_i' and v_j' (in V') where e is negative-incident to both v_i' and v_j'.



(a) MAX-CUT Instance (b) MAX-DIR Construction

Figure 6: An instance of the MAX-CUT problem shown with the reduction to MAX-DIR.

Clearly the construction takes polynomial time.

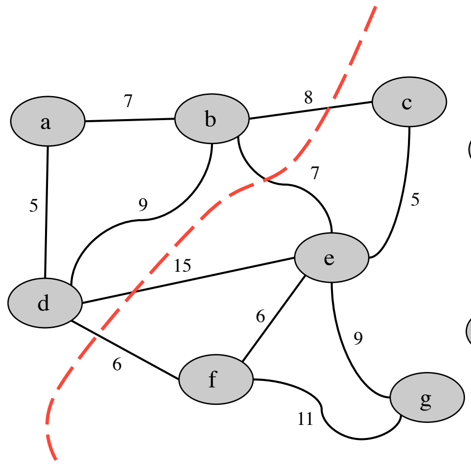
First we show that if M has a cut of size k, then G has a vertex-orientation assignment yielding k directed edges. If M has a cut of size k (e.g., Figure 7a), then there is a partition of V into two distinct subsets S and T such that there are k edges which have an endpoint in S and an endpoint in T. Note that by partitioning V' into the same subsets, S and T, and assigning forward-orientation to all vertices in S and reverse-orientation to all vertices in T, k bidirected edges (those analogous to the cut edges of M) are rendered directed edges (see Figure 7b). All others remain either introverted or extraverted edges. It follows from the same line of reasoning that if G has a vertex-orientation assignment yielding k directed edges, then M has a cut of size k.

This completes the proof that MAX-DIR is NP-complete.

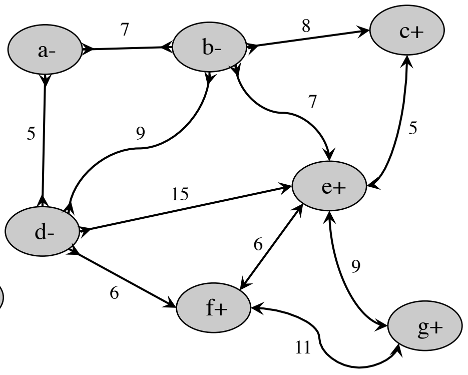
3.2 Corollary

MAX-DIR $\leq_m^P$ *MAX-CUT*.

Because MAX-CUT is NP-complete and MAX-DIR$\in$NP, it follows naturally that this corollary is true. In fact, for any two NP-complete problems, A and B, it will always be true that $A \leq_m^P B$ and $B \leq_m^P A$. An alternative way to express this is to say that A is many-one equivalent in polynomial time to B (i.e., $A \equiv_m^P B$). The polynomial-time reduction function from MAX-DIR to MAX-CUT is slightly more involved than the reduction from MAX-CUT to MAX-DIR, but the details are beyond the scope of this manuscript. Suffice it to say that such a function does exist, as proven by the corollary.



(a) An Example Cut



(b) Corresponding Vertex-Orientation Assignment

Figure 7: A possible cut of the multigraph in 6a and the corresponding vertex-orientation assignment of 6b. Note that both the weight of the cut and the sum weight of the directed edges are the same.

4 Discussion

The proof that MAX-DIR is NP-complete is useful because it signals that a heuristic will be required to solve an instance of MAX-DIR on any reasonably large input. The corollary is useful because it allows us to reduce any MAX-DIR to a MAX-CUT problem. Thus, rather than "reinventing the wheel" to solve instances of the MAX-DIR problem, we can apply existing MAX-CUT heuristics (of which there are many good ones) to obtain efficient, accurate solutions.

In my research I have applied an efficient MAX-CUT heuristic to several instances of the MAX-DIR problem. My findings suggested that in fact a simple greedy algorithm performs better than the heuristic due to the particular nature of the graphs being studied. Thus, being able to apply existing algorithms may not always be advantageous, even when such algorithms perform exceptionally well in other contexts.

References

[1] George M Church, Yuan Gao, and Sriram Kosuri. Next-generation digital information storage in dna. *Science*, 337(6102):1628–1628, 2012.

[2] Jack Edmonds and Ellis L Johnson. Matching: A well-solved class of integer linear programs. In *Combinatorial structures and their applications (Gordon and Breach*. Citeseer, 1970.

[3] Mihai Pop, Daniel S Kosack, and Steven L Salzberg. Hierarchical scaffolding with bambus. *Genome research*, 14(1):149–159, 2004.