Supplementary Materials for TransPS: A Transcriptome Post Scaffolding Method for Assembling High Quality Contigs

Mingming Liu, Zach N Adelman, Kevin M Myles, and Liqing Zhang

Supplementary Method

The TransPS follows an align-layout-consensus procedure. It consists of three major stages. First, original contigs were used to search for their best alignments to the reference amino acid sequence database using BLASTX. Second, contigs are placed in the right order and orientation according to their aligned coordinates to the reference. Third, those nonredundant contigs matching to the same reference sequences are scaffolded/assembled into one supercontig. Let T_i be the set of contigs that match with the same reference protein i. We divided T_i into three subsets, contigs that are accepted as final contig and need no further processing (A_i), contigs that are redundant (R_i), and contigs that are used to scaffold into one supercontig (S_i).

In the alignment stage, BLASTX is used to guide the alignment of contig sequence. Specifically, contig sequences are used as queries against a protein sequence database and the best target sequence, i.e. the sequence with the lowest e-value and highest bit score, is selected as the best match to the query. It is possible that multiple cintigs match the same protein sequence due to incomplete assembly of sequences, which is referred to as a one-to-many matching, otherwise a one-to-one matching. In the case of one-to-many matching, all the contigs matching the same protein are connected and put into a set of sequences T_i, where i represents protein i with which all contigs in T_i match.

To produce a high quality set of contigs, it is important to remove redundant sequences generated by the assemblers. The contigs in T_i are divided into three subsets, contigs that are accepted as final contig and need no further processing (A_i), contigs that are redundant (R_i), and contigs that are used to scaffold into one supercontig (S_i). The one-to-one matching contig goes to set A if it matches at its positive strand. For one-to-many matching, sequences in T_i are assigned to R_i, A_i or S_i. If two contigs match the same protein at the same region or overlap with each other for a certain percentage (user defined argument), then one of the two contigs is considered to be redundant and thus moved to R_i. After removing the redundant and accepted sequences, all the remaining sequences in T_i go to subset S_i for scaffolding, except in the case of only one contig left for scaffolding, it goes to A_i subset. In the following two stages, we focus on the contigs for scaffolding in S_i.

In the layout stage, the purpose is to order the contigs in S_i based on the reference coordinates

provided by the alignment. Intuitively, to ensure the consistency with the reference genome, the contigs are sorted by their matching start positions on the reference protein in ascending order.

In the consensus stage, the purpose is to scaffold the original contigs in S_i from s_1 to s_n $s_i \in S_i$ $(1 \leq i \leq n)$ into a single high quality contig based on the order obtained from the previous stage. The most important part for scaffolding is to estimate the "scaffolding parts", which are determined based on two cases, overlapped contigs or separated contigs. The contigs in S_i are scaffolded into one supercontig in the consensus stage as discussed in detail below.

Each sequence in the scaffold group S_i is typically associated with four coordinates, which are the matching begin (b_i) and end (e_i) position of the reference sequence, and the matching begin (b'_i) and end (e'_i) position of the query sequence. We always have $b_i \leq b_{i+1}$ because of the setting in the layout stage. The algorithm is shown in Algorithm 1. We discuss it based on two cases.

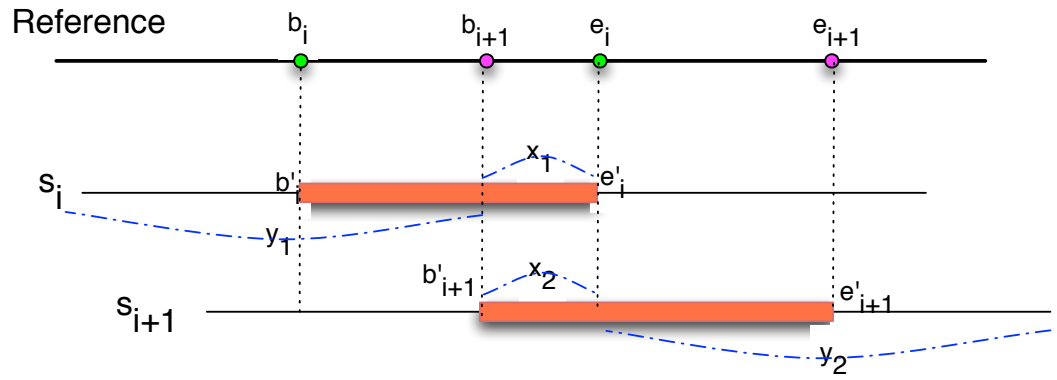
Algorithm 1 Scaffolding Algorithm

place contigs in S (scaffolding group) in order according to their match starting position (b_i) against the reference protein.
$s = s_0$, $b = b_0$, $e = e_0$, $b' = b'_0$, $e' = e'_0$
$i = 0$ //$s_i \in S$
while $i < |S| - 1$ **do**
  $s_i = s$, $b_i = b$, $e_i = e$, $b'_i = b'$, $e'_i = e'$
  $x_1 = s_i[e'_i - (e_i - b_{i+1}), e'_i]$ //assume $e_i > b_{i+1}$
  $x_2 = s_{i+1}[b'_{i+1}, b'_{i+1} + (e_i - b_{i+1})]$ //assume $e_i > b_{i+1}$,
  $jointSeq = \text{LCS}(x_1, x_2)$; //Longest Common String
  $s = s_i[1, e'_i - (e_i - b_{i+1})] + jointSeq + s_{i+1}[b'_{i+1} + (e_i - b_{i+1}), n_{i+1}]$
  //n_{i+1} is the length of contig s_{i+1}
  $b = b_i$
  $e = b + (b_{i+1} - b_i) + |jointSeq| + (e_{i+1} - e_i)$
  $b' = b'_i$
  $e' = e'_i - (e_i - b_{i+1}) + |jointSeq| + (e_{i+1} - e_i)$
  $i = i + 1$
return c

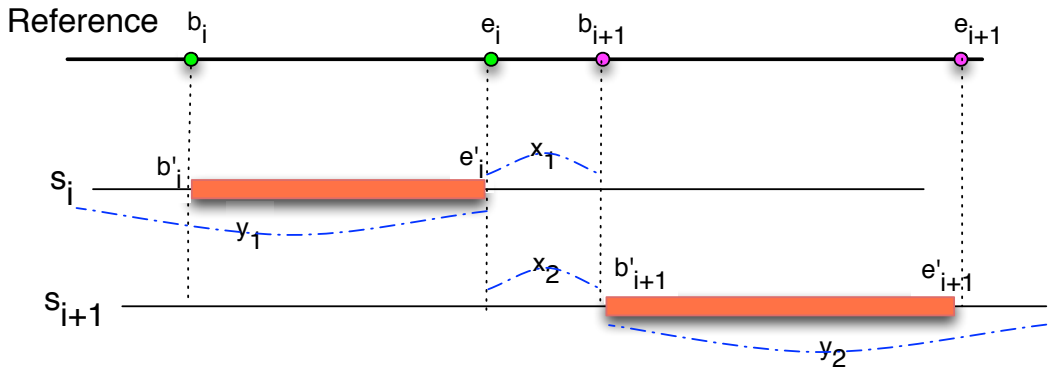
Case I: Overlapping

The matching intervals of two contigs s_i ($[b_i, e_i]$) and s_{i+1} ($[b_{i+1}, e_{i+1}]$) against the same reference sequence overlap. That is $e_i > b_{i+1}$ as shown in Figure S1 (a). Two sequence segments, x_1 and x_2 , are the overlapping region of s_i and s_{i+1}. Let y_1 be the subsequence from the beginning of s_i to the position right before x_1; y_2 be the subsequence starts right after x_2 to the end of s_{i+1}. The sequence joining y_1 and y_2 is defined as the scaffolding part f, which is obtained based on the longest common

substring (LCS) of x_1 and x_2. For example, $x_1 = ATGAACT$ and $x_2 = CAACTCA$. Obviously, the LCS of x_1 and x_2 is $AACT$, thus $f = ATGAACTCA$. As a result, we link y_1, f and y_2 together as a new contig and remove s_i and s_{i+1} from S_i. Note that to avoid short tandem repeats, we make sure that f is obtained beginning with base from x_1 and ending with base from x_2. The reading frame also needs to be considered. For example, in the reverse condition of the last example, $x_1 = CAACTCAT$ and $x_2 = ATGAACT$, we obtained $f = AACT$. However, the length of f is not divisible by 3, which results in a frame shift in the original sequence, we add N to the end of f to ensure the sequence is in frame, which leads to $f = AACTNN$.



(a)



(b)

Figure S1: Two cases considered in the scaffolding procedure. (a) Case I: Matching intervals ($[b_i, e_i]$ and $[b_{i+1}, e_{i+1}]$) overlap. (b) Case II: Matching intervals ($[b'_i, e'_i]$ and $[b'_{i+1}, e'_{i+1}]$) do not overlap. x_1, x_2, y_1, and y_2 represent substring of s_i or s_{i+1}.

Case II: Nonoverlapping

The matching intervals of two contigs s_i ($[b_i, e_i]$) and s_{i+1} ($[b_{i+1}, e_{i+1}]$) against the same reference sequence do not overlap. That is $e_i < b_{i+1}$ as shown in Figure S1 (b). The same as in Figure S1 (a), x_1,

x_2, y_1 and y_2 are marked in Figure S1 (b). If there are nucleotide bases at the right end of s_i and at the left end of s_{i+1} (typically ≥ 10 bp), f is obtained the same as in case I. However, if there are no bases at either end, for example, e'_i is the end of sequence s_i, a extension strategy of x_1 and x_2 is used by setting $e_i = e_i - \epsilon$, $e'_i = e'_i - \epsilon$, $b_{i+1} = b_{i+1} + \epsilon$, and $b'_i = b'_i - \epsilon$. We set $\epsilon = \max(0, r * (b_{i+1} - e_i - \min(n_1, n_2)))$, where r is a user defined extension ratio ($0 < r \leq 1$, $r = 0.1$ by default), $n_1 = \text{length}(s_i) - e'_i$, and n_2 (length(s_i) returns the number of nucleotides of contig). Then calculate f based on the new coordinates. Finally, as in case I, y_1, f, and y_2 are linked together as a new contig, and s_i and s_{i+1} are removed from S_i.

The procedure is recursively implemented based on the above two cases until all the sequences in S_i are merged into one contig or removed.