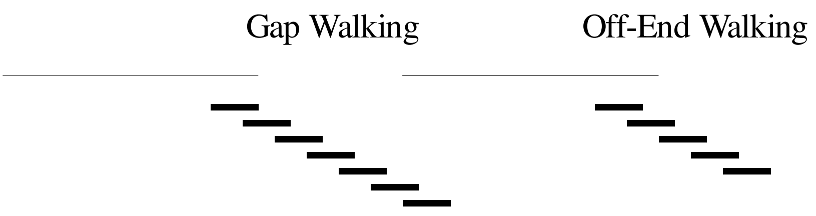


External BAC Walker

1. Overview

The goal of the External BAC Walker (EBW) is to utilize the information available via BACs to augment the scaffold. There are several cases where BAC information can potentially be used to add coverage to an assembly. One case is when a gap occurs in the interior of a locale, and a BAC exists that spans the gap. Another case is at the end of a contig where the BAC could be used to extend coverage into the gap or off the end of a scaffold.

The basic approach to use BAC information in either case is to identify the BAC fragments that are near the end of the contig, and attempt to use contigs containing the appropriate BAC fragments to either walk the gap between contigs or to extend off the end of the contig.

The EBW will run as part of CGW. Thus we will utilize as much as possible the existing mechanisms and data structures in this effort.

2. Basic Design

The EBW will apply the following approach to each scaffold:

a. Scan the scaffold a gap at a time.
b. Identify the BACs that have fragments within a certain distance of the end of the contigs that flank the gap.
c. If any BAC has fragments on both sides of the gap, attempt to walk the gap using only contigs that contain fragments from that BAC. If more than one BAC has fragments on both sides, choose the BAC to use based on the agreement between the estimated gap size and the distance between the BAC fragments.
d. If no BAC has fragments on both sides of the gaps (or we are at the end of a scaffold), select BACs which extend into the gap and attempt to walk out into the gap utilizing contigs which contain fragments from that BAC.

The current gap walking module has many of the features that will be needed for BAC walking. However, there are differences in the data and algorithm between the two approaches:

- i. the current gap walker assumes that all the overlap edges among the chunks in the graph are present,
- ii. the current gap walker takes only the quality of the overlap, and does not take into consideration distance relationships present in the underlying fragments that the EBW wants to utilize,
- iii. the current gap walker is not designed to walk off the end of contigs,
- iv. the current Frag Store does not allow for easy retrieval of all fragments associated with a locale,
- v. the current gap walker includes all the available chunks in the graph it constructs to search for a walk.

Item (i.) can be addressed by calculating all the needed overlap scores. This would be accomplished by identifying all the chunks that contain fragments from the desired locale and calculating the overlaps based on the positions of the contained fragments in the original locale. I.e., we will use the positions of the BAC fragments in the locale to determine which overlap relationships are induced and calculate only those overlaps.

Item (ii.) will be addressed by constructing an appropriate edge scoring function for the gap walking. For instance, the scoring function can be weighted to favor edges where the overlap between the chunks is consistent with the positions of the contained fragments in the locale.

Item (iii.) will require a new approach. One possible method would be to walk much as in current walking, adding chunks whose overlaps and contained locale chunks are consistent and checking to make sure that the placement of these chunks are consistent with other placed data.

Item (iv.) is being addressed in the design of the *Assembler Grande* Frag Store. In the current design there is an index structure that allows for the retrieval of all fragments associated with a locale.

Item (v.) will be addressed by constructing a new subgraph-building routine that limits the chunks in the subgraph to those in the chosen BAC or BACs.

Item (v.)

3. Plan

Items (i.) and (iii.) will require the most effort. The functionality of item (i.) can be incorporated into the current gap walking code as part of a new routine to build the graph that the walking occurs over. The overlap scores can then be stored inside existing data structures and accessed during the walking.

Item (iii.) may be able to be addressed via a new termination condition for the current walker. Currently, a successful walk ends when the destination chunk is reached and the walk distance is consistent with the gap estimate. In the case of BAC walking off of the end, the termination condition could, e.g., be reaching the last of the chunks that contain fragments from the correct locale, or when no chunk that contains the correct locale fragments are present.

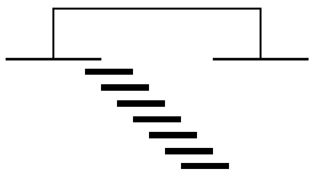
The functionality needed to deal with item (ii.) can be achieved through the construction of the appropriate scoring function. The edges available off of a particular node will already be highly constrained due to the way the graph is constructed. The weighting function will have to decide among only the edges that are consistent with the currently planned 3x shredding of the data.

4. Cases and Priorities

There are a number of difficult cases beside the relatively straightforward instances of inserting unscaffolded chunks to fill a gap or the extension of a contig out into a gap or off the end of a scaffold. In this section we illustrate some of those cases, and give an order in which we plan to handle such cases.

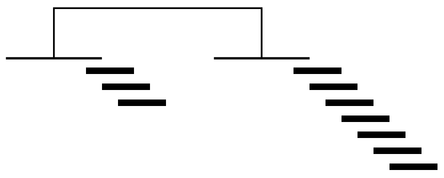
Case 1 is the normal gap walking mode, i.e., there is a gap spanned by fragments from a BAC and we use the unscaffolded chunk containing those fragments to walk the gap.

Case 1



Case 2 is the next easiest case to handle. Here we walk off the end of a contig, either because we could find no spanning locale, or we are at the end of a scaffold.

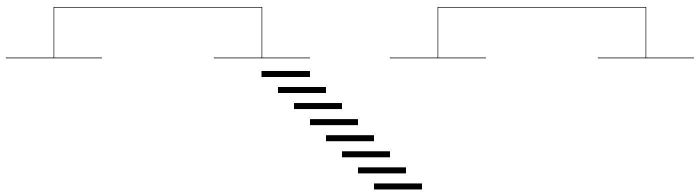
Case 2



Cases 3, 4, and 5 all involve locales that have fragments in two or more scaffolds. Some mechanisms have already been developed to deal with walks involving two scaffolds. In the current walker, if a scaffold is completely contained within the gap of another, the smaller scaffold is merged with the larger. (See Case 4.) One possibility for dealing with interleaved scaffolds (Case 5) is to mark the one scaffold dead and to use the chunks from that scaffold as needed to walk or fill gaps in the other.

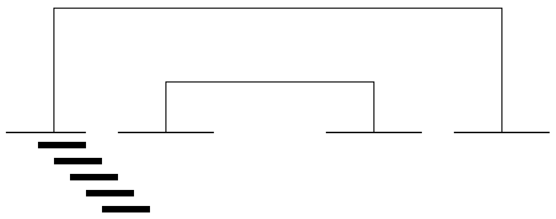
In Case 3, there are fragments from a single locale that are contained in contigs from two separate scaffolds.

Case 3



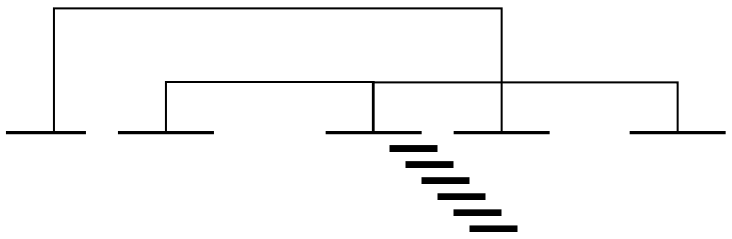
In Case 4, a smaller scaffold is completely contained in the gap of a larger scaffold. The fragments from the locale may not span the gap.

Case 4



Finally, in Case 5, the scaffolds are interleaved. This case is considered rather difficult to deal with.

Case 5



5. Needed Infrastructure

Locale indices
CIFragT fields

Authors:
Created: Mike Flanigan
Date: 01/26/00