Local Unitigging for Repeat Resolution

Karin A. Remington

1. Overview

The proposed repeat resolution strategy is to consider each gap in turn , or a set of neighboring gaps, gathering any material that potentially belongs in a given gap (by virtue of mate links to proximal contigs), and apply unitigging rules in an attempt to congeal the data, thereby closing or narrowing the gap.

2. Process

Borrowing from the rocks and stones code base in AS_REZ, potential fill material can be determined for each gap within a scaffold. An all-to-all overlap computation is performed, resulting in an overlap graph on this (presumably small) set of primarily local data. It's hoped that the localization of the data will enable unitigging to successfully find meta-unitig(s) which close or narrow the gap.

3. Issues

Separate process, or wrapped into CGW? NEEDS TO BE WRAPPED INTO CGW… and early in the process so that gaps are narrowed *before* scaffold merging has the opportunity to make any mistakes.

One neighborhood? Or greater than 1? PERHAPS ONE NEIGHBORHOOD TO START, with successive iterations?

To get the overlap edges, do we need to go back to the overlap store? Would there have been edges lost in the upstream process that we might want to recapture? NO OVERLAP EDGES

Keep unitigs as is, or split into component fragments and re-unitig them? GRANGER's INITIAL THOUGHT IS TO USE UNITIGS, this requires fragment instantiation…

How to perform the all-to-all overlap:

- ❑ DP_Compare on unitigs? (for overlaps, this isn't too efficent, since some of the banding advantage is lost). But has the advantage that it works out "out of the box" on unitig length objects.
- ❑ Overlapper?

How to implement the graph and the unitigging rules:

- ❑ LEDA would be easier to program, but might be difficult to integrate within the Assembly build process. (I think IMPOSSIBLE to integrate is closer to the truth.)
- ❑ Could implement simple unitigging with simple graph structures a la current CGW
- ❑ Functional interface to unitigger, might be best from an engineering point of view, to preserve a single codebase and automatically inherit all improvements made in AS_CGB code. ISSUE about how much current unitigger directly or implicitly thinks in terms of fragments versus unitigs.

4. Feasibility Study

It would be instructive to look at some examples from simulations and components or grande human data and see what might happen by using unitigger out-of-the-box on the fragments that are considered potential fill in a gap

AUTHORS

Karin A. Remington

`Created August 31, 2001`

`Modified September 7, 2001`