Repeat Resolution: Preliminary Specification & Design

Art. Delcher

4 November, 1999 Version 1.13

1. Overview

The Repeat Resolution Module of the Assembly Subsystem has the task of determining the positions in the assembled genome of those unitigs that the Unitigger module was unable to characterize as unique. The assigned positions will be relative to the coordinates assigned for scaffolds by the Chunk Graph Walker module. The position could be within a single chunk of scaffolds, or between 2 (or more?) separate chunks of scaffolds.

2. General Strategy

The approach taken by the Repeat Resolution Module will be to apply a series of decreasingly rigorous tests to each non-unique unitig in an attempt to assign it (or various subsets of its fragments) a position relative to the chunks of scaffolds previously assembled by the Chunk Graph Walker.

Based on our simulation experience, we expect that most unitigs will, in fact, contain fragments from only a single region of the genome. Thus, the initial tests on a unitig will regard it as an atomic element. Only when there is evidence that a unitig is not from a single region of the genome, attempts be made to break it apart.

There are two types of evidence to determine the position of a unitig: overlaps and mate pairs. Of these, we generally expect mate pair evidence to be more reliable.

In general, our approach can be regarded as "chunk-centric" in that it starts with the unitigs and attempts to find a place for each one. An alternative approach could be "gap-centric" and attempt to fill in the gaps in the scaffolds. The gap-centric approach may be a more natural way to approach the higher-numbered tasks in the following list.

Currently Repeat Resolution is organized into 4 levels:

- ♦ Rez I—"Rocks". In this level we attempt to insert into scaffolds only unitigs that have at least 2 consistent mate links that specify their position and which pass additional consistency checks. This is the most conservative insertion stage and should make very few errors, if any.
- ♦ Rez II—"Stone-Throwing". Here we try to insert unitigs that have only 1 mate link or possibly conflicting mate links. To confirm the positions of these insertions we find a path of overlapping contigs through these stones. The insertions made in this stage are more likely to be erroneous. Accordingly, to avoid further placements based on previous errors, we currently do not iterate this stage.

Note that since some unitigs may be "overcompressed", *i.e.*, they include fragments from different portions of the genome, we allow the same unitig to be placed in multiple gaps in this steps. What is placed initially is just a "surrogate" for the unitig, consisting of just the consensus sequence for it. In the Rez IV step below we will distribute the fragments for each unitigs to its surrogates.

- ♦ Rez III—"Gap Walking". This level performs a greedy, quality-based overlap walk across gaps in the scaffold, filling in the unitigs through which it passes. This stage should have a similar error rate to Stone-Throwing and is not iterated. As above, only surrogates are inserted in this stage.
- ♦ Rez IV—"Fragment Placement". This is the final stage in which all remaining fragments are assigned to their best-guess position in the scaffolds.

The top-level algorithm for applying these is as follows:

```
do
{
  Insert Rocks;
  Recompute Scaffold Positions;
}  while  (Number of Rocks Inserted >= Threshold);
Insert Stones;
Do Gap Walking;
Place Remaining Fragments;
```

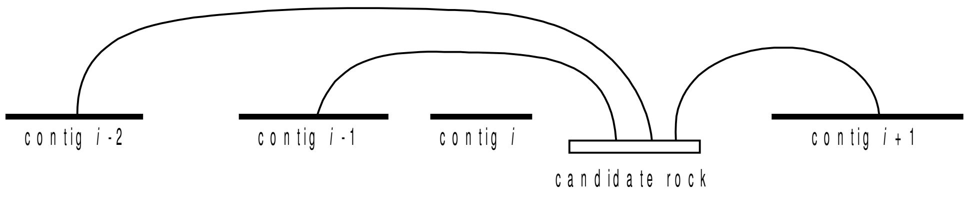
3. Rez I—Rocks

3.1.Initial assignment of unitigs to gaps

The first unitigs that we attempt to assign to scaffold gaps are those with edge mates to scaffolded contigs that incorporate 2 or more fragment mate links. Specifically, for each unscaffolded unitig, or *candidate rock*, we do the following:

1. Check its arrival-rate statistic. If this value is below the threshold specified in `MIN_ROCK_COVER_STAT`, skip further processing of this unitig in this round.
2. Examine its edge mates to scaffolded contigs. We discount any edge that does not include a mate link, or has been flagged as `probablyBogus`, or is a BAC-end guide link (which is too long to be useful for placement purposes).
3. If the selected edge mates collectively incorporate at least 2 fragment mate links, and they all connect to the same scaffold, and the connections all imply the same orientation of this chunk, then continue to step 4. We also currently allow a single mate link to another scaffold if there at least 3 mate links to the same scaffold.
4. Now calculate the end positions (relative to the scaffold coordinate frame) implied by the selected edge mates. This calculation is based on assuming that the position variance of the nearest-to-the-left scaffold contig is 0, and then adjusting other variances relative to this. We do this to avoid large accumulated variances near the end of a scaffold.

For example, consider the following situation:



The candidate rock has edge mates (indicated by arcs) to 3 of the 4 scaffold unitigs shown. Let (m_{i-2}, v_{i-2}) be the mean and variance of the right end of unique i –2, (m_{i-1}, v_{i-1}) be the mean and variance of the right end of unique i –1, and (m_{i+1}, v_{i+1}) be the mean and variance of the left end of unique i +1, as computed by the chunk graph walker.
We first adjust the variances to make

$$v'_{i-2} = v'_{i-1} - v'_{i-2}$$
$$v'_{i+1} = v'_{i+1} - v'_{i-1}$$
$$v'_{i-1} = 0$$

We then compute an estimate on the left and right positions of the candidate rock for each of the 3 edges:

- ♦ For the edge to unique i – 2 we estimate the mean of the left position of the rock to be m_{i-2} + (mean length of the edge). The variance of the left estimate is v'_{i-2} + (length variance of the edge).
  The estimate for the right position is the same, but with the mean and variance of the length of the non-unique chunk added.
- ♦ Likewise, for the edge to unique i – 1 we estimate the left mean of the rock to be m_{i-1} + (mean length of the edge) and the variance to be v'_{i-1} + (length variance of the edge).
  The estimate for the right position is the same, but with the mean and variance of the length of the non-unique chunk added.
- ♦ For the edge to unique i + 1 we estimate the right mean of the rock to be m_{i+1} - (mean length of the edge) and the variance to be v'_{i+1} + (length variance of the edge).
  The estimate for the left position is the same, but with the mean length of the non-unique chunk subtracted and the length variance added.

Finally the estimates from the separate edges are combined into a single estimate with mean

$$m = \Sigma\,(m_j / v_j) \,/\, \Sigma\,(1 / v_j)$$

and variance

$$v = 1 \,/\, \Sigma\,(1 / v_j)$$

5. Next the candidate rock is assigned to a scaffold gap. The assignment is to the

nearest gap based on distances from the gap midpoints to the rock's midpoint. Note that we treat the region before the first unitig in the scaffold and after the last as gaps and will assign unitigs to them.

6. If necessary, the positions of the ends of the rock are recomputed so that the reference variance is the chunk immediately to the left of the assigned gap (or to the right for the gap before the scaffold). This is done to ensure that all rocks assigned to the same gap have a consistent reference position. For example, in the above diagram, the positions would be recalculated assuming a zero variance for the right end of unitig *i*.
7. Once positions (mean and variance) are calculated for the ends of the candidate rock, these values are tested for consistency with the edges from which they were derived. The test is described in the next section. If the test fails, the candidate rock is eliminated from further processing.
8. We have implemented an additional, more stringent check on unitigs. This reduces the number of unitigs placed into scaffolds, but fits our philosophy of making the most conservative moves first. We can do the less conservative step in subsequent iterations.
   The additional test is that each unitig to be placed in a gap have no link-mate edges to any contig except those in the scaffold where it's being inserted or other unitigs being inserted in this round. In preliminary runs, this test avoids the insertion of a few "invalid" unitigs. Currently this additional test can be selected by a command-line option.

3.2. Testing rocks for consistency.

After having assigned a number of candidate rocks to each gap in a scaffold, we check whether the above-calculated positions (called "assumed positions") are mutually consistent. For each rock, we check its consistency with every other rock in two phases. If the first phase is passed, we trust the assumed positions. If the first phase was not applicable or failed we go to the second phase. If this test fails also, we reject the assumed positions for this round. The two phases work as follows:

1. In the first phase we employ a distribution test that uses in essence the same formulas with which the assumed positions were calculated. If there is an edge mate between two rocks, it implies a distribution on the gap length between them. In addition we estimate the gap length using the assumed positions.
   In terms of the following picture:

The distribution based on the assumed positions is

(m_e,v_e) = (S_2.mean-E_1.mean,E_1.variance+S_2.variance).

This is combined with the distribution (m_g,v_g) of G given by the edge mate to give a combined distribution (m,v) with mean

$$m = (m_g/v_g+m_e/v_e) / (1/v_g+1/v_e)$$

and variance

$$v = 1 / (1/v_e+1/v_e)$$

Then we intersect the ±3 standard-deviation intervals around the mean of the combined distribution with the same interval of both the estimated and the given distribution. If both intersections are non-empty, we accept the rock. If one of them is empty, we pass it to phase two.

2. In phase two we employ an overlap test that works as follows:
   For every rock we check which other rocks in the same gap it supposedly overlaps. This is done by estimating the gap distribution as explained above and checking whether the mean of the gap indicates a sufficiently large overlap (currently the overlap mean must be greater than 0). Then we explicitly compute the overlap of the two rocks and compare the computed overlap with the indicated overlap. We return success (that means that the two rocks overlap) if the relative error

   | (*computed overlap – indicated overlap*) / *indicated overlap* |

   is smaller than some threshold.

   If all overlap tests are passed then we accept the positioning of the tested rock, otherwise we reject it unless the following situation occurs.
   If one overlap test fails it might be that the *other* rock (the one not being tested) is positioned badly. To confirm this assumption we require that the assumed badly placed rock does NOT overlap with all rocks that overlap the tested one. If this is the case we keep the tested rock (the badly placed one will be rejected in it's test).

3.3.Additional Checks

In the current version, the positions of rocks are confirmed by searching for an overlap path in exactly the same fashion as for stones, as described below.

4. Rez II—Stone Throwing

The second level of repeat resolution involves placing unitigs which have only a single mate link connecting them to scaffolded chunks, or which have conflicting mate links to scaffolded chunks. We think of such unitigs as "stones" thrown from the scaffold into gaps. If the placement of the stone is confirmed by overlap information, then it can be added to the scaffold.

Every unitig with a clone-mate link to a scaffolded chunk is tentatively assigned the scaffold position indicated by the mate link. Note that the same chunk may be tentatively assigned to multiple positions.

Specifically, each unitig not yet in a scaffold is considered. If it has no mate links to scaffolded chunks, it is eliminated from further consideration. Its mate link edges are then clustered according to the position they imply for the unitig. (Currently, the clustering is done naively based on the scaffold to which a connection is implied. We need to add a step that allows multiple placements within the same scaffold. This becomes increasingly important as scaffolds get larger.) From each cluster of edges, a tentative position for this unitig is calculated, exactly the same way as for rocks above. For each such position, a "stone" representing that position is assigned to the nearest gap in the scaffolds.

After all stones have been assigned a tentative position in scaffold gaps, we attempt to find an overlapping path of contigs that confirms the positions of as many of the stones as possible. The current implementation of this path finding is a simple, depth-first search, starting from the contig on one end of the scaffold gap, trying to reach the contig on the other end of the gap. The search ignores edges which are suspect. (This currently includes those marked as part of tandem repeats and those that imply a containment relationship.) The position of each contig found in the search is the first one encountered in the depth-first visit. From the dag implied by these positions, the path that:

- hits the most stones,
- in positions consistent with their tentatively calculated positions,
- and reaches the goal (the other end of the gap)

is chosen.
The stones on this path are then regarded as confirmed, and their positions are adjusted to be compatible with the positions on the path. The position of the goal of the path also is adjusted, to avoid conflicting with the stone positions. These confirmed stones are then inserted into scaffolds.

5. Rez III—Gap Walking

The third and last level of repeat resolution tries to greedily close remaining gaps. Momentarily gap walking tries to close gaps within scaffolds (intra-scaffold gaps). Basically we employ a DFS-traversal of the chunk graph using overlap edges which tries to connect two nodes by a walk. At any node we choose the edge that has the best quality according to some quality function. If we find the destination we check whether the implied length of the walk is consistent with the gap estimate given by mate link information. If so we unwind the path of best edges and close the gap by inserting the respective surrogate chunks into the scaffold.

After stone throwing and a final merging of scaffolds we sort the existing scaffolds according to their size. Then we try to close gap on a per scaffold basis starting with the biggest scaffold.

For each scaffold we scan it and collect all gaps in it. Then we estimate the gap length using the mate link information already present in the scaffold (i.e. we estimate it by the difference of the scaffold positions of the flanking chunks).
Using this estimate we try greedily to find a path from the left chunk (in A to B direction in the scaffold) to the right chunk.

- For each node in the walk we look at a subset of its outgoing edges (e.g. no containment overlaps).
- We sort this subset according to a bayesian quality function. 0.0 is the best quality whereas 1.0 is bad (repetitive overlap).

Then we recursively call the path finding algorithm for these edges in sorted order of quality. If we are returned a successful walk we stop exploring the edges and return. If we do not find a successful walk exploring this edge we go to the next edges in the sorted list. At any level the exploration of the list stops if

  a) we have explored a certain number of adjacent edges which is a walk parameter
  b) the quality of the edge is above a certain threshold

- In addition we give up on exploring the gap as soon as the overall number of explored edges exceeds a certain global threshold.
- Whenever we reach the destination chunk we check whether the resulting walk lies within an interval of X (X =3 or 5) standard deviations around the mean of the gap estimate. If not we deem the walk as unsuccessful and continue in the exploration of the edges.

After having walked all gaps in a scaffold we compute their positions in the scaffold and adapt the gap estimates accordingly. Then we insert them in the scaffold. A chunk is not inserted in the scaffold if

- if it is in another scaffolds and the other chunks in this scaffold are not on the walk (this is conservative and could be changed)

that means potentially there can be gaps in a walked gap. In general a surrogate of the chunk is inserted, except for the case that the chunk was previously in a scaffold. In this case it is not split into a surrogate chunk before inserting.

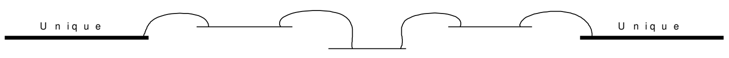
After the chunks are inserted a contigging phase for this scaffold follows and we proceed to the next scaffold.

6. Rez IV—Fragment Placement

7. Other moves

The following are other criteria for selecting unitigs to place into or between scaffolds, listed in approximate order of reliability:

- ♦ Unitigs that meet the criteria of section 4.1, except that they would be placed on the end of two different scaffolds. Such unitigs are strong evidence that the scaffolds should be merged. All these unitigs would be computed and their scaffold merge implications checked for consistency, *i.e.*, if some unitigs indicated the same end of unitig *i* should merge with the unitig *j* while others indicated it should merge with unitig *k*, then neither merge would be made (at this point).
- ♦ Unitigs with "confirmed" link-mate connections to a particular scaffold position. For example, the links shown in the following diagram



are confirmed by the known distance (implied by some other link or by guides) between the two unique unitigs. We can find such paths in two ways:

1. By finding paths from one unitig to the other. Chunks on the path could be placed into the gap. The quality of edges used in the path can be varied. Paths containing many confirmed link edges are preferable to those containing just overlaps.
2. By finding biconnected components in the graph. The biconnectedness guarantees that each unitig has two or more connections that specify its position. The biconnected components that contain a unique unitig can have the other unitigs checked for consistency and then placed relative to the unique(s).

- ♦ For each unplaced chunk, we can compute the first unique that can be reached by pursuing a path of edges out from each of its ends. If only one unique can be reached from either end, that is strong evidence for the placement of the chunk. Even if more than one unique is reachable, the set of reachable uniques can be used to classify the unplaced chunk into a repeat category.

8. Implementation Notes

We plan to begin development of repeat resolution as a function within the Chunk Graph Walker. Should it become difficult to synchronize development within a single module, we can split it off to a separate module, but until that becomes necessary it will speed development to be able to use the data structures already available with the Chunk Graph Walker.

AUTHORS

Art. Delcher:

Initial:	10 May 99
Revised:	25 May 99
	More details added to original GapFiller.rtf .
Revised:	12 Jul 99
	Added descriptions of other moves.
Revised:	29 Jul 99
	Revised section 4.1.
Revised:	23 Nov 99
	Expanded section on stones.