Modification : Transitive edges removed from each node

Function : SGTransitiveReductionVisitor

Class : assemble.cpp

Parameters : graph

This uses Myers' algorithm(2005).

Modification : Remove contained vertices from the graph

Function : SGContainRemoveVisitor

Class : assemble.cpp

Parameters : graph

Contained vertices are removed from the graph by remodeling the graph. The remodeling of the graph is done if the graph has been transitively reduced, we have to check all the neighbors to see if any new edges need to be added. If the graph is a complete overlap graph we can just remove the edges to the deletion vertex

Modification : Validate that the graph does not contain any extra edges or missing irreducible edges

Function : SGValidateStructureVisitor

Class : assemble.cpp

Parameters : graph

Returns or prints if vertices have no irreducible edges or extra irreducible edges

Modification : Remodel the graph to infer missing edges or remove erroneous edges

Function : SGSmallRepeatResolveVisitor

Class : assemble.cpp

Parameters : graph, minDiff

If a vertex has more than MAX\_EDGES(10) and less than 2 edges, we do not attempt to resolve else we resolve in the following way :

i. We eliminate the shortest edge from the vertex to be removed.

For eg. Let there be an edge X-->Y from X but if Y has a longer edge than Y->X in the same direction then we remove X- ->Y.

ii. Delete the edge if the difference between the shortest and longest is greater than minDiff

~~ACGTT~~

X Y

CTTAAGTTC

Modification : Detects and removes small "tip" vertices from the graph when they are less than minLength in size

Function : SGTrimVisitor

Class : assemble.cpp

Parameters : graph, minLength, num\_island, num\_terminal(dead end)

Nodes that either do not have edges or edges are in one direction are considered as "island" and "dead end" vertices respectively. Returns no of such vertices.

If an island, remove if the sequence length is less than the threshold(minLength)

Modification : Smooth out variation in the graph

Function : SGSmoothingVisitor

Class : assemble.cpp

Parameters : m\_simpleBubblesRemoved, m\_complexBubblesRemoved, m\_numRemovedTotal

1. findVariantWalks(pVertex, dir, MAX\_DISTANCE, MAX\_WALKS, variantWalks) MAX\_DISTANCE = 5000 MAX\_WALKS = 10
2. Calculate the walk coverage using the internal vertices of the walk. The walk with the highest coverage will be retained
3. Check the divergence of the other walks to this walk. We want to compute the total gap length, total mismatches and percent divergence between the two paths. matchLen, totalDiff, gapLength, maxGapLength
4. The vertex set for each walk is not necessarily disjoint, the selected walk may contain vertices that are part of other paths. This is also handled.
5. if(variantWalks.size() == 2 m\_simpleBubblesRemoved += 1; else m\_complexBubblesRemoved += 1;

Modification : Remove identical vertices from the graph

Function : SGIdenticalRemoveVisitor

Class :

Parameters : graph

Contained vertices with same sequence and equal length are removed.

This algorithm is less complex than SGContainRemoveVisitor because we do not have to remodel the graph in this case because no irreducible edges need to be moved.

Modification : Remove edges from the graph when the ratio between an edge's overlap length and the longest overlap for the vertex is less than the given parameter

Function : SGOverlapRatioVisitor

Class :

Parameters : minRatio

Remove edges from the graph when the ratio between an edge's overlap length and the longest overlap for the vertex is less than the given parameter

Modification : Detect and remove duplicate edges

Function : SGDuplicateVisitor

Class :

Parameters : graph

Duplicate edges are removed

Modification : Remove the edges of super-repetitive vertices in the graph

Function : SGSuperRepeatVisitor

Class : assemble.cpp

Parameters : m\_num\_superrepeats

Repeated vertices are those which has greater no of edges than the specified MAX\_EDGES(10) parameter.