ScaffoldGraph Design

This document describes the ScaffoldGraph data structure that is intended to support scaffold construction, repeat resolution, contig construction, and output of the Chunk Graph Walker component. The ScaffoldGraph encodes all of the information in the ChunkGraph and the current ChunkOfScaffoldsGraph in a form that is supposed to be easy to manipulate in the late phases of the Chunk Graph Walker.

This data structure incorporates 4 different views of the extended chunk graph (all terms should be defined below):

1) Graph of Chunk Instances (GCI)
2) Graph of Contigs (GC) – Overlapping Chunk Instances are merged into Contigs, and the edges between CIs are „bundled“ into edges between Contigs
3) Graph of Scaffolds of Contigs – Contigs are placed in scaffolds and the Contig Edges are further bundled into edges between Scaffolds.

In all three graphs, the edges in the graph represent relations between the positions and orientations of pairs of entities, that are induced by the overlap computed by the unitigger, and the mate/guide links. The edges between discriminator unique chunk instances are an invariant. All other edges between CIs may change as a function of decisions to split (and join?) chunk instances during the CGW's computation. Likewise, the edges between contigs and scaffolds are a function of the decisions made to group chunk instances into contigs and scaffolds. As such, recomputing the edges in the graph is a key operation. The three views of the data **are not** automatically maintained in synchrony.

Definitions:

Chunk Graph: A graph whose vertices are chunks as defined by the unitigger, and whose edges reflect a merging of both mate links and overlaps, and a separate merging of guide links and join contraints.

Chunk: A collection of reads, with associated relative positions, as defined by the unitigger. A node in the ChunkGraph. Each chunk has an arbitrarily assigned ‚A‘ and and ‚B‘ end, and each fragment within the chunk has a relative orientation (A_B or B_A).

EdgeMate: : An edge in the chunk graph that specifies a relative orientation of two chunks (same notation as in the overlap graph), as well as an estimate of the gap (positive distance) or overlap (negative distance) between the chunks. The EdgeMates are derived from one or more of the following sources:

- An overlap between two chunks in the ChunkGraph
- A pair of fragment mates in the two chunks
- A pair of guide mates in the two chunks
- A pair of join mates in the two chunks

Chunk Instance (CI): A node in the GCI view of the ScaffoldGraph. A subset of the reads associated with a unitigger chunk or chunks, along with their associated mate edges. For unique chunks, there is a single chunk instance for each chunk. For repeat chunks, the fragments and EdgeMates associated with the unitigger generated chunk initially reside in an Unresolved Chunk, and are eventually split into 2 or more subsets, and associated with different parts of the assembly. The following types of ChunkInstance are defined:

- DiscriminatorUniqueChunk – Initially, only CIs with this label are scaffolded
- UnresolvedChunk - The initial state for all chunks that are not discriminator unique

- ResolvedUniqueChunk
- ResolvedRepeatChunk
- Contig

Chunk Instance Edge (CIEdge): A relationship between two chunk instances, based on an EdgeMate in the ChunkGraph, between a pair of chunk instances. These are expressed as undirected edges, with each ChunkInstanceEdge linked into lists such that it is simple to enumerate all ChunkInstanceEdges emanating from a particular ChunkInstance.

Contig: A node in the GC view of the ScaffoldGraph. Chunk Instances that are contiguous within a scaffold may be replaced by a contig, that collects within it all of the CIs fragments. When Contig edges are constructed, the CIEdges of the underlying CIs (except those that involve two CIs within the same contig) will induce Contig edges.

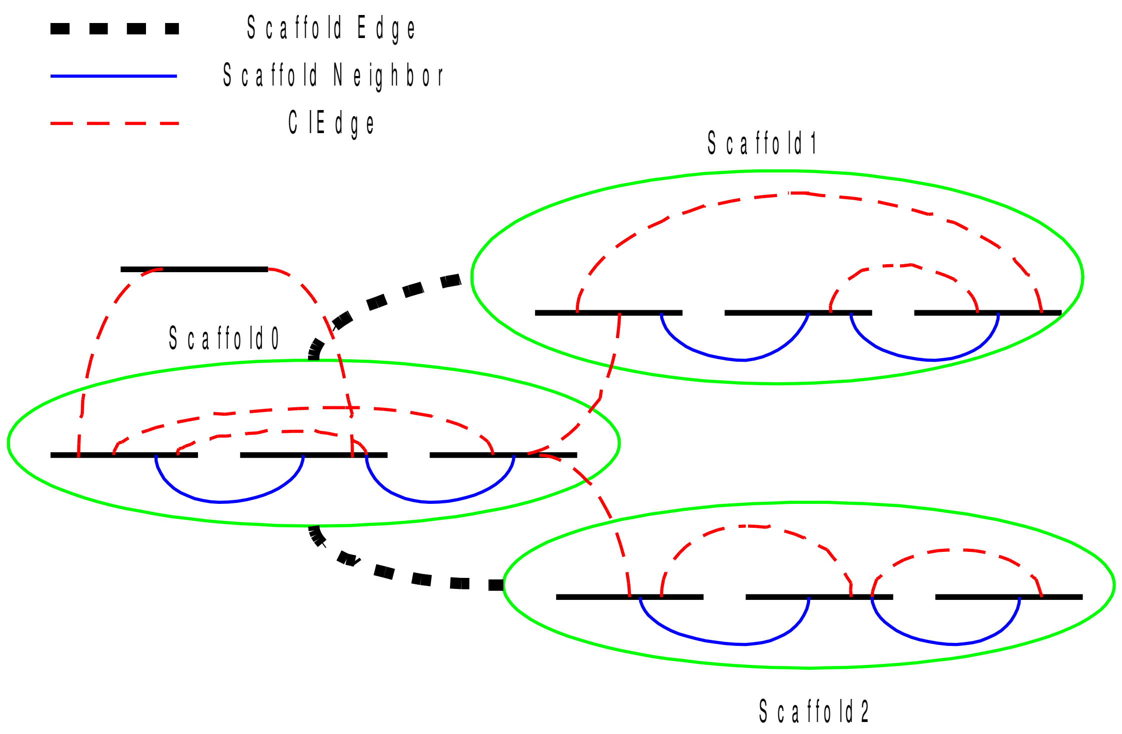
Contig Edge : A relationship between two contigs, based on an EdgeMate in the GC, between a pair of contigs. These are expressed as undirected edges, with each ChunkInstanceEdge linked into lists such that it is simple to enumerate all ChunkInstanceEdges emanating from a particular ChunkInstance.

Scaffold: A node in the GSC and GSCI views. An ordered collection of Chunk Instances, with associated orientations and relative positions, that represent a single path through the Chunk Graph. Where scaffolds branch due to large polymorphisms, scaffold edges will be present. Initially, Scaffolds contain only instances of discriminator-unique chunks. Repeat resolution will add single instances of unique chunks that were not classified as discriminator-unique, as well as multiple instances of repeat chunks.

ScaffoldEdge: An edge in the GSC and GSCI views. A relationship between two Scaffolds, based on one or more EdgeMate in the ChunkGraph that has been associated with CIs in the two scaffolds. These are expressed as undirected edges, with each ScaffoldEdge linked into lists such that it is simple to enumerate all ScaffoldEdges emanating from a particular Scaffold.

Scaffold Graph: The sum of the graphs of Chunk Instances, Contigs, and Scaffolds.

Assembly: An unordered collection of scaffolds.



Assumptions

1) Fragments that are assigned to multiple chunks can be safely ignored
2) Fragments will be linked together so that all fragments belonging to a particular chunk will be in a single linked list.

General Remarks

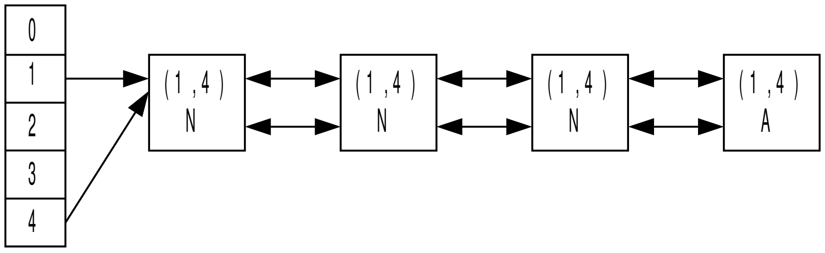
1) All references are stored as indices of elements in an array.
2) All memory is allocated from variable length arrays

Graph Edges

All graph edges (edges between CIs, Contigs and Scaffolds) operate in the same manner, and use much of the same machinery. An edge between elements (I,j) will appear in the sorted doubly-linked edge lists of both element I and element j. Edges are undirected, and are stored in canonical form such that the index of the first node is less than the index of the 2nd node. Let us illustrate with an example. Given four nodes, with indices 1-4, and the following edges:

(1,4)	Normal orientation
(1,4)	Normal orientation
(1,4)	Normal orientation
(1,4)	Antinormal orientation

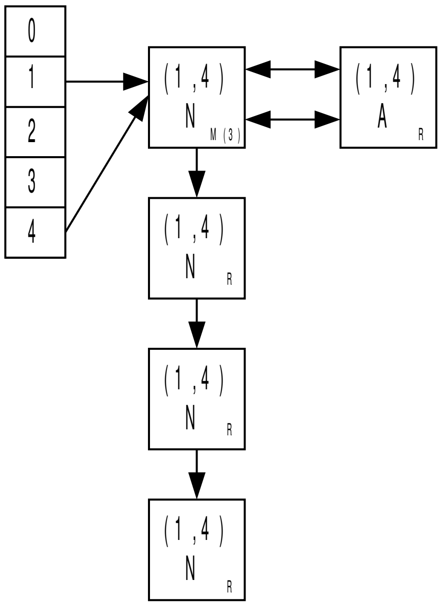
The following graph of the raw edges would result:



a) Edges Prior to Merging

The double-ended arrows between the edges represent the 4 links per edge that support their inclusion in the doubly-linked edge lists of two nodes.

Assuming the 3 (1,4)N edges are statistically compatible, the following edges would result:



b) Edges Following Merging

The merged edge (marked ‚M') reflects the mean and variances of the raw edges. The raw edges are attached to the merged edges, and can easily be accessed.

Constructing the Scaffold Graph

The scaffold graph is constructed directly from the COSGraph and the ChunkGraph. Ideally, the Scaffold graph would be constructed directly, and the COSGraph could be retired. Initially the Scaffold Graph contains a CI for every chunk in the ChunkGraph, but only the unique scaffold seeds from the COSGraph are „Scaffolded".

CIs are constructed directly from their counterparts in the Chunk Graph. Scaffolds are created, and CIs are inserted in scaffolds. CI Edges and scaffold edges are constructed and inserted into the Scaffold Graph.

Operations

Operations on Chunk Instances

- CreateInitialChunkInstance Create a chunk instance from a Chunk graph chunk
- CreateSplitChunkInstance Create a Chunk Instance by splitting an existing Chunk Instance. This performs all of the mechanics of splitting the CI – partitioning its fragments and CIEdges – according to the parameters that are passed.
- GetScaffoldPredecessor Get the index of the scaffold neighbor of a particular CI in the direction of the 'A' end of the scaffold.
- GetScaffoldSuccessor Get the index of the scaffold neighbor of a particular CI in the direction of the 'B' end of the scaffold.

Operations on CIEdges

- FindCIEdge
- InsertCIEdge
- ExtractCIEdge
- Iterate over all CIEdges of a particular CI
- Iterate over all CIEdges between CIs I and j

Operations on Scaffolds

- CreateCIScaffold Create a new CI Scaffold
- MergeScaffolds Merge two scaffolds into one
- InsertCIInScaffold Insert a chunk instance in a scaffold
- RemoveCIFromScaffold Remove a CI from a scaffold
- RecomputeOffsetsInScaffold Recompute the offsets of the CIs in a scaffold
- ReplaceCIsWithContig Replace a series of CIs with a contig
- Iterate over all CIs in a Scaffold

Operations on Scaffold Edges

- FindScaffoldEdge
- InsertScaffoldEdge
- ExtractScaffoldEdge
- Iterate over all ScaffoldEdges of a particular scaffolds
- Iterate over all ScaffoldEdges between scaffolds I and j

Operations on Scaffold Graph

- RebuildScaffoldEdges Construct and Merge the scaffold edges following addition of CIs to scaffolds, and/or other rearrangements of the Scaffold Graph.

Implementation Details

Memory Allocation

All objects are allocated from one of several VarArrays. The collection of these VarArrays is contained within a ScaffoldGraphT structure.

All references are int32 indicies.

Cledges and Scaffold Edges

These links are stored as undirected edges, linked together to facilitate a traversal of all edges incident on a particular vertex.

Scaffolds

The CIs within a Scaffold are maintained in a doubly-linked list. The Scaffold object maintains a reference to the two CIs at the A and B ends of the Scaffold.

AUTHORS

Saul Kravitz :
`Created May 17, 1999`