From phased facts to gfa graph

Raw phased allele

DiscoSnp++ or DiscoRad can outputs a file named `phased_alleles_read_set_id_X.txt` per input read set (`X` varies from `1` to `N`). This is obtained thanks to the hidden option `-A` of `run_discoSnp++.sh` and `run_discoSnpRad.sh`
This file indicates which SNP alleles overlap in which direction and with which nucleotide numbers between bubble sequences. It also outputs the number of reads that validated this overlap.

```
-1000610l_0;-2184581l_22;-5049471h_10;6970552l_4;-1621389l_1;344859l_2;763097h_
1; => 5
```

Here, lower path of reverse `1000610` is followed by lower path of reverse `2184581`. The two bubble sequences are separated by 22 nucleotides.

```
---XXXXXXXXXXXXXXXX---------------------------  1000610
         --------------------------XXXXXXXXXXXXXXX------------ 2184581
                   <--- 22 nuc. -->
```

And so on. The 7 alleles shown on this example are validated by 5 reads.

A run of such phased alleles is called a **fact**.

Special case of paired reads

If disco was run with paired files, the `phased_alleles_read_set_id_X.txt` is as follow:

```
9994l_0;13575l_132; -31756l_0;30683h_37;-29126l_31;-23618l_59; => 12
```

The line has a space, and so two facts are represented. They correspond to the facts obtained thanks to two reads.
The distance between the two facts is unknown.
We call this a **paired fact**.

Creation of a graph from raw phased alleles.

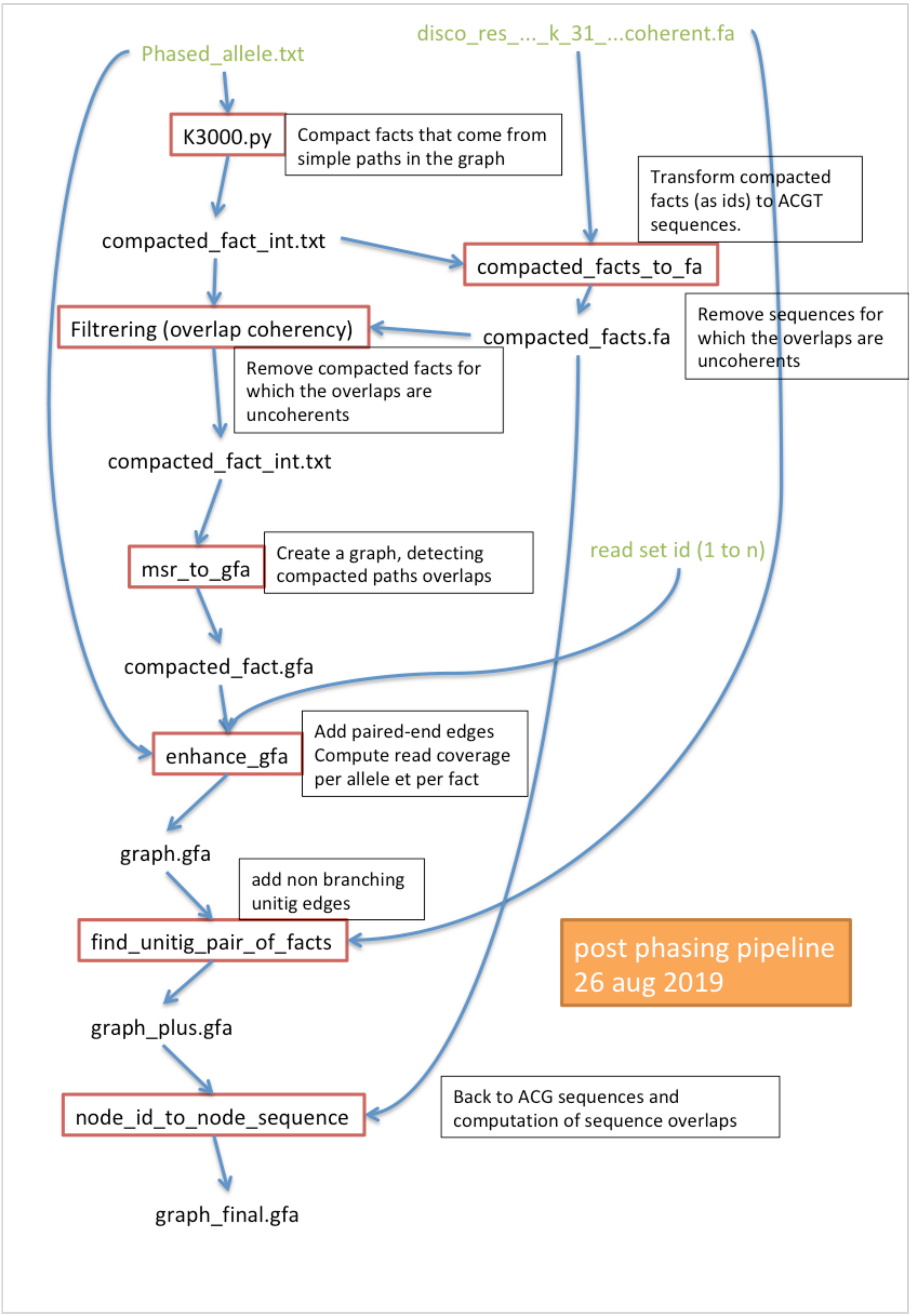
We propose to create a graph from those phased facts. This consists in several steps:

1. Detect overlaps between facts (edges)
2. Compact facts that come from simple paths in the graph
3. Find back ACTG sequences corresponding to compacted fact ids.
4. Add Particular edges:
    1. Edges corresponding to paired links between two facts
    2. Edges obtained thanks to unitig left and right extensions of SNPs produced by disco.
5. Compute for each fact:
    1. Its coverage as obtained by the number of reads that **phase** alleles of a fact. This is in the `FC` field of gfa files.
    2. Its coverage as obtained by the minimal read coverage among all alleles of a fact. This is in the `RC` field of gfa files.
6. Put back sequences in the gfa file
7. Compute the real sequence overlap between nodes (removing self looping nodes and nodes included in other ones)

All scripts are located in the `scripts/k3000` directory of discoSnp.
They can be run at once by running

```
run.sh phased_alleles_read_set_id_1.txt
discoRes_k_31_c_2_D_0_P_3_b_2_coherent.fa
```

We whole pipeline is scketched here:


Phased_allele.txt
disco_res_..._k_31_...coherent.fa
K3000.py
Compact facts that come from simple paths in the graph
Transform compacted facts (as ids) to ACGT sequences.
compacted_fact_int.txt
compacted_facts_to_fa
Filtrering (overlap coherency)
compacted_facts.fa
Remove sequences for which the overlaps are uncoherents
Remove compacted facts for which the overlaps are uncoherents
compacted_fact_int.txt
msr_to_gfa
Create a graph, detecting compacted paths overlaps
read set id (1 to n)
compacted_fact.gfa
enhance_gfa
Add paired-end edges Compute read coverage per allele et per fact
graph.gfa
add non branching unitig edges
find_unitig_pair_of_facts
post phasing pipeline 26 aug 2019
graph_plus.gfa
node_id_to_node_sequence
Back to ACG sequences and computation of sequence overlaps
graph_final.gfa

Step 2. Compact facts that come from simple paths in the graph

Actually the process starts with the point 2. (even if this step needs the computation of overlaps, they are re-computed later).

```
python3 K3000.py  phased_alleles_read_set_id_1.txt >  compacted_facts_int.txt
```

The two first steps are performed simultaneously by the `K3000.py` script.
The output is a set of compacted facts (edges are lost at the end of this step).
A compacted fact is on the form:

```
38772;-21479;27388;-495;65526;29404;34837;-13757;
```

For historical technical questions, we need only integer values for the process. Thus alleles are transformed with the following bijection.

```
19386h -> 38772
-10739l -> -21479
```

- From x_path (with path in {h,l}) to the int-encoded allele:

```
x_path -> 2x+0 if path = h
          2x+1 if path = l
```

In the `compacted_facts_int.txt` file we loose the distance between alleles and we loose the paired information.

Step 3. Find back ACTG sequences corresponding to compacted fact ids.

This step is obtained thanks to the command:

```
python3 /Users/ppeterlo/workspace/gatb-discosnp/scripts/k3000/
K3000_compacted_paths_to_fa.py discoRes_k_31_c_2_D_0_P_3_b_2_coherent.fa
compacted_facts_int.txt > compacted_facts.fa
```

This steps outputs a fa file, in which headers contain int-encoded facts and the starting-ending position of each fact in the final sequence. The sequence is the corresponding ACGT sequences, concatenation of the (overlapping) sequences of the facts.
During this step some compacted facts are removed as their sequences do not overlap correctly (indel in mapped reads for instance)

Step 1 (at least). Detect overlaps between facts (edges)

```
python3 K3000_msr_to_gfa.py compacted_facts_int.txt > compacted_facts.gfa
```

From the previously created `compacted_facts_int.txt`, we generate the first `.gfa` file.
In this file, the int-encoded alleles are re-transformed into the `x_path` representation.
Each compacted fact is now a node, a line starting with an `S` followed by an id, and the corresponding phased allele compacted fact.

```
S 9   19386h;-10739l;13694h;-247l;32763h;14702h;17418l;-6878l;
```

Edges are lines starting with al `L`. Edges are oriented

```
L 846 +   9   +   4M
```

Here the compacted fact 846 (forward) overlaps the compacted fact 9 (forward) with 4 alleles.
Verification:
Compacted fact `846`

```
S 846 3151h;-19607l;-14005l;19386h;-10739l;13694h;-247l;
```

Overlap: between compacted fact `846` and compacted fact `9` with an overlap of `4` alleles:

```
3151h;-19607l;-14005l;19386h;-10739l;13694h;-247l;
                      19386h;-10739l;13694h;-247l;32763h;14702h;17418l;-6878l;
```

Step 4. 1 Edges corresponding to paired links between two facts

```
python3 enhance_gfa.py compacted_facts.gfa phased_alleles_read_set_id_1.txt>
graph.gfa
```

This adds edges, simply indicating that two compacted-facts happened in paired reads:

```
L 9961+   20467   +   0M  FC:i:109
```

Each time a paired-fact `f1 f2` perfectly maps two compacted facts (`f1` maps compacted fact `cfA` and `f2` maps compacted fact `cfB`, we count an edge `L   cfA  + cfB + 0M`. The total count (here `109`) is the sum of the number of paired-facts that link `cfA` to `cfB`.

warning we do not check to orientation of such alleles. They are always `+ +`. Checking orientation is a feature that has to be implemented.

Step 4.2 Edges obtained thanks to unitig left and right extensions of SNPs produced by disco.

This step is obtained thanks to unitigs from disco. Option `-t` is thus mandatory.

```
python3 find_unitig_connected_pairs_of_facts.py graph.gfa
disco_k31_..._coherent.fa > graph_plus.gfa
```

This script determines the `k` value from the file name that must have a `_kXX` somewhere in its name.
We derive from unitigs of SNPs the pairs of SNPs that are consecutive on the genome and that have no branching between them. In this case, SNP `A` is followed by SNP `B` iif

- Last k-1-mer of the right unitig of `A` is equal to the last k-1-mer before the SNP position (excluded) of B.
- First k-1-mer of the left unitig of `B` is equal to the first k-1-mer afeter the SNP position (excluded) of A.

We detect those conditions for SNPs that are at the extremities of each compacted facts.
This adds new edges linking compacted fact:

```
L 19946   +   11433   +   -1M
```

They are oriented edges, and they all finish with -1M in order to differentiate them from paired edges.

Step 5. Compute fact coverages.

This step is also obtained during the `enhance_gfa.py` phase.

1. Coverage as obtained by the number of reads that **phase** alleles of a fact. This is in the `FC` field of gfa files.

We obtain this result by summing for each compacted fact the coverage of each facts (obtained from raw_phasing files (eg ` => 12`) that belong to this compacted fact.

2. Its coverage as obtained by the minimal read coverage among all alleles of a fact. This

is in the `RC` field of gfa files.
We obtain this result by considering the read coverage (from the raw disco `.fa` file) of each allele that compose a compacted fact. Note: we also dispose from the max and mean values, that are non used currently.

Steps 6 and 7. Put back sequences in the gfa file :

This step is obtained with :

```
python3 K3000_node_ids_to_node_sequences.py graph_plus.gfa compacted_facts.fa >
graph_final.gfa
```

The `graph_final.gfa` file contains ACGT sequences instead of fact ids.
The overlap size of edges with positive values are the real overlap size of the corresponding sequences

TODOs (22/08/2019):

- [ ] Orientation of paired edges
- [ ] Usage of contigs from discoSnp (-T) ➜ unitigs may be deducted from the header
- [x] ~~Nodes: transform allele ids to ACGT sequence~~
- [ ] Better implementation of `L` lines in `modify_gfa_file` of `K3000_node_ids_to_node_sequences.py`. Use starting and ending positions of facts in compacted facts to retreive the size of the overlaps instead of re-computing sequences overlaps.
- [ ] Determine maximal flows to reconstruct haplotypes from uneven covered haplotypes (metaG, polyploids, RAD, ...)