Repeat resolving using paired data

Dmitry Antipov and Alexander Sirotkin

June 8, 2011

Abstract

In this paper we give a general definition for validness of repeat resolving methods based on information, obtained from mate pairs. We introduce a class of operations on graphs that saves all possibilities to reconstruct genome and prove that the rectangle graph technique belongs to this class in case of error-free data. Furthermore, we provide some modifications of that technique which belongs to this class and provides better results than rectangle graph. In the last section we introduce an adaptation of our technique to non-errorless data.

1 Motivation

Most of sequencing machines provide not single reads, but mate-reads. We will show that mate-pair information can be used in more efficient way than modern assemblers do. It seems that right place for mate-pair information usage is repeat resolving step. But to discuss different methods of repeat resolving based on paired information usage, we need to define what the correct repeat resolving method is and understand how we can compare quality different repeat resolving methods. In terms of graph the genome can be represented as a cycle (or path in case of non-cyclic genome). While applying different graph simplification methods we need to preserve this cycle.

2 About paired information

Assume that we have a directed graph $G = (V, E)$ with specified lengths of all its edges. G can be any abstract graph, but in case of genome assembling you may think that G is compressed[1] de Bruijn graph. For given edge a we denote it's length as l_a.

Remark 1. *In case of compressed de Bruijn graph with k-mers as vertices, length of edge is the number of $(k + 1)$-mers in this edge.*

Definition 1. *Directed distance between two vertices A and B in path P, started in A and finished in B, is the sum of the lengths of all edges in path P.*

---

[1] Formal definition of compressing operation will be given later. Informally, for each vertex with only one incoming edge and only one outgoing edge we remove this vertex and concatenate this edges.

Definition 2. *Directed distance between two edges* a *and* b *in path* P *started from edge* a *and finished by* b *is sum of the lengths of all edges of* P *with exception of* b.

For each edge e, we define $\text{start}(e)$ and $\text{end}(e)$ as start and end vertex of edge e respectively. Length of path P is sum of length of its edges (multiple occurrences allowed). We denote it as $|P|$. For such graph we define *edge pair* as ordered pair (a, b) where $a \in E$ and $b \in E$. *Edge pair info* is a pair $((a, b), x)$ where (a, b) is edge pair, and $x \in \mathbb{N}$.

Definition 3. *For a given constant* d *and path (or cycle)* P *(from directed graph* G*), pair info* $((a, b), x)$ *is called* d-consistent edge pair info with respect to P *if there exists directed subpath* P' *of* P, *such that* P' *starts from* a, *ends by* b, x *is distance between* a *and* b *in* P' *and following inequalities hold[2]:* $x - l_a \leq d$, $x + l_b \geq d$.

Definition 4. d-set of edge pair info *is set* S *of edge pair infos such that there exists a Chinese postman cycle* C *(possibly not unique) such that* $((a, b), x) \in S \Leftrightarrow ((a, b), x)$ *is* d-consistent with respect to cycle C.

2.1   Receiving paired info from reads

It seems to be a subject of another paper...

3   Repeat resolving

We want to define what is "correct" graph transformation with respect to paired information and provide methods for comparison of these graph transformations in terms of the resulting graph complexity.

3.1   Ungluings

Definition 5. *For given graph* $G = (V, E)$, *we say that graph* $G' = (V', E')$ *is an* ungluing *of* G *if there exists surjective mappings* $f : V' \to V$ *and* $g : E' \to E$, f *and* g *such that:*

1. *Edge length in* G' *must be consistent with* g *i.e.* $\forall a \in E', l_a = l_{g(a)}$.

2. $\forall e' \in E'$ $f(\text{start}(e')) = \text{start}(g(e'))$ *i.e.* $f \circ \text{start} = \text{start} \circ g$

3. $\forall e' \in E'$ $f(\text{end}(e')) = \text{end}(g(e'))$ *i.e.* $f \circ \text{end} = \text{end} \circ g$

---

[2]This condition guaranty us that in uncompressed graph there was a pair of edges with lengths one, with distance exactly d between them, such that first edge was compressed into a and second into b.

Remark 2. *For every* ungluing G' *of* G *we can define gluing operation in terms of [Pevzner 2004] such that* G *can be obtained as the result of performing this gluing operation to* G'. *This operation glues* $f^{-1}(a)$ *into* a *for all* $a \in V$, *and* $g^{-1}(e)$ *into* e *for all* $e \in E$.

Definition 6. Compress operation *for graph* G *consists of following steps:*

1. *Select a vertex* A *with in-degree and out-degree both equal to one.*

2. *Remove the uniquely defined edges* a *and* b, *with* $\text{end}(a) = A$ *and* $\text{start}(b) = A$.

3. *Remove vertex* A.

4. *Add edge* $(\text{start}(a), \text{end}(b))$.

Definition 7. *For a given graph* $G = (V, E)$, *we say that a graph* $G'' = (V'', E'')$ *is a* compressing *of* G *if* G'' *is obtained from* G *by any number of compress operation.*

Remark 3. *For any edge from compressing* G'' *of* G *we can find path in* G *that is compressed into this edge.*

Definition 8. *For a given graph* $G = (V, E)$, *we say that a graph* $G'' = (V'', E'')$ *is a* compressed ungluing *of* G *if there exists an ungluing* G' *of* G, *and* G'' *is compressing of* G'.

Till the end of this subsection, let $G = (V, E)$ be a graph, $G' = (V', E')$ — its ungluing and $G'' = (V'', E'')$ — compressed ungluing of G.

Definition 9. *For a given directed path* $P' = (e'_0, \ldots, e'_n)$ *in* G' *we define* decoding *of this path as* $(g(e'_0), \ldots, g(e'_n))$ *and denote it as* $\text{dec}(P')$.

Definition 10. *For a given* d-*set* S *of edge pair info and a cycle* C *from graph* G, C *is a* valid *cycle iff* $((a, b), x) \in S \leftrightarrow ((a, b), x)$ *is* d-*consistent with respect to cycle* C.

Definition 11. *We say that cycle* C' *from graph* G' *is* valid *for a given* d-*set* S *of edge pair info in graph* G *if* $\text{dec}(C') = C$ *and* C *is valid cycle in* G.

Definition 12. *For a given* d-*set* S *of edge pair info in* G, *if* $\text{dec}(\cdot)$ *is a bijection between sets of valid cycles in* G *and* G', *we'll say that ungluing* G' *is* S-*consistent ungluing.*

Definition 13. *For given* d-*set* S *of edge pair info, if* G'' *is a compressing of any* S-*consistent ungluing of* G, *we'll say that* G'' *is* S-*consistent compressed ungluing.*

Remark 4. *Note that "ungluing" is a graph notion that in general is independent on pair info. An ungluing* G' *can be* S-*consistent with respect to one* d-*set* S *of paired info and not* S-*consistent with respect to other one.*

3.2 Informal reformulation

Every valid cycle in G corresponds to a variant of genome, consistent with graph and paired information. So, our condition means that any S-consistent ungluing is a transformation of G which preserves all possibilities for genome from G and does not produce any new possible genome cycles.

Remark 5. *The above definitions can be simply reformulated for linear genomes.*

3.3 Graph complexity estimation - still under discussion!

Problem 1. *With given two S-consistent ungluings of a graph G, we want to define which one is better.*

Let's consider an ungluing $G' = (V', E')$ of graph $G = (V, E)$. For every vertex $v' \in V'$, we'll define *vertex incoming-outgoing set* (and denote it as $O_{v'}$) as $\{(a, b) | a \in E, b \in E\}$: (a, b) satisfies following conditions:

1. $\exists a' \in E' : end(a') = v', g(a') = a$

2. $\exists b' \in E' : start(b') = v', g(b') = b$

We call vertex *simple* if both in-degree and out-degree of this vertex both equal to one. Let's denote the set of simple vertices of graph G' by $U_{G'}$. *Graph incoming-outgoing set* is $\bigcup\limits_{v' \in E' \setminus U_{G'}} O_{v'}$. *Measure of complexity* for G' is the size of G' graph incoming-outgoing set. The complexity of any compressing of G is equal to complexity of G.

There are many other possible measures, for example, minus sum of all edges length.

4 Connection with rectangle graph

Definition 14. Vertex pair info *is a triple* (A, B, x), *where* A *and* B *are vertices and* $x \in \mathbb{Z}$

Definition 15. Vertex-edge pair info *is a triple* (A, b, x), *where* A *is vertex,* b *is edge and* $x \in \mathbb{Z}$

Let's consider rectangle graph[3] approach to repeat resolving. There are three steps in it (without less of generality, we'll contract blue edges):

1. Building all possible rectangles.

2. Gluing vertices with same labels.

3. Contracting blue edges.

---

[3]We assume that basic concept of rectangle graph is known by reader.

Let's denote the resulting graph after all three steps as $G_r = (V_r, E_r)$ and result of first two steps as $G_{rb} = (V_{rb}, E_{rb})$.

Remark 6. *Labels of vertices in rectangle graph can be considered as vertex pair info.*

Each vertex v_{rb} from V_{rb} has a label — vertex pair info $(A(v_{rb}), B(v_{rb}), x(v_{rb}))$. We call first element of such triple *first mark* of vertex v_{rb} and denote it $fm(v_{rb})$.

Since contracting of blue edges is a gluing operation, there exists a partition of V_{rb} into family of disjoint subsets $\{H_{rb}(v_r)\}_{v_r \in V_r}$ such that all vertices from $H_{rb}(v_r)$ are glued into v_r. For a given vertex $v_r \in V_r$, let's consider $H_{rb}(v_r)$. Since two vertices from G_{rb} connected by blue edge have same first mark, we can define mapping $f : V_r \to V$ by following equation:

$$f(v_r) = fm(v_{rb}), \text{ where } v_{rb} \text{ any element of } H_{rb}(v_r).$$

Each edge of G_r has label from E by construction. So this labeling provides us the mapping $g : E_r \to E$.

The conditions on f and g from definition of ungluing are satisfied by construction of rectangle graph.

Remark 7. *By construction, every vertex in one blue-connected component (set of vertices connected by blue edges) of G_{rb} has the same vertex $V \in G$ as first mark.*

Remark 8. *The third step (contracting blue edges) is equivalent to constructing a graph with blue-connected components of G_{rb} as vertex set. Two vertices in the new graph are connected with edge an labeled $e \in E$ iff in the corresponding connected components of G_{rb} there are two vertices connected with an edge labeled e.*

Definition 16. *For given graph $G = (V, E)$ and d-set of edge pair info P we say that vertex pair info (A, B, x) is* obtained *from P if there exists edge pair info $((a, b), y)$ such that:*

- *a incoming edge for A, b incoming edge for B and $x = y - l_a + l_b$,*

- *a outgoing edge for A, b incoming edge for B and $x = y + l_b$,*

- *a incoming edge for A, b outgoing edge for B and $x = y - l_a$,*

- *a outgoing edge for A, b outgoing edge for B and $x = y$.*

Definition 17. *For given graph $G = (V, E)$ and d-set of edge pair info P we say that vertex-edge pair info (A, b, x) is* obtained *from P if there exists edge pair info $((a, b), y)$ such that:*

- *a is an incoming edge for A and $x = y - l_a$,*

5

- a *is an outgoing edge for* A *and* $x = y$.

We denote such edge pair info $((a, b), y)$ (possibly not unique) as a *precursor* of the vertex-edge pair info (A, b, x).

Definition 18. *Two vertex-edge pairs* (V, a, x) *and* (V, b, y) *are called* adjacent *if* a *and* b *have at least one common[4] vertex in graph* G *and distances from* V *to common vertices of* a *and* b, *obtained from* (V, a, x) *and from* (V, b, y), *are the same.*

We denote set of all vertex-edge pair infos obtained from P as P_v.

$$P_v = \{(A, b, x) | A \in E, \exists (a, b, y) \in P : (a, b, y) \text{ is precursor for } (A, b, x)\}$$

For vertex $Z \in E$, $P_v(Z) = \{(A, b, x) | (A, b, x) \in P_v; A = Z\}$.

Now we can define *rectangle split operation*. This operation split one vertex at time. If we start from any graph G, and apply this operation consecutively we will obtain G' that is S-consistent ungluing of G.

Definition 19. *For a given vertex* $Z \in V$ *rectangle split operation consists of the following steps:*

1. *Partition* $P_v(Z)$ *into equivalence classes* C_i *by the transitive closure of the adjacency relationship so that:* $P_v(Z) = \bigcup C_i$.

2. *For each class* C_i, *create a vertex* Z_i.

3. *For each precursor* $((a, b), y)$ *of each vertex-edge pair info* $(Z, b, x) \in C_i$, *create a copy* a_i *of the edge* a *replacing*

   - $\text{start}(a)$ *with* Z_i *iff* $\text{start}(a) = Z$ *and vertex-edge pair* (Z, b, x), *obtained from* $((a, b), y)$, *belongs to* C_i.
   - $\text{end}(a)$ *with* Z_i *iff* $\text{end}(a) = Z$ *and vertex-edge pair* (Z, b, x), *obtained from* $((a, b), y)$, *belongs to* C_i.

4. *For each precursor* $((a, b), y)$ *of each vertex-edge pair info* $(Z, b, x) \in C_i$, *replace* $((a, b), y)$ *in* P *with* $((a_i, b), y)$.

5. *Remove* Z *with all its incoming and outgoing edges.*

Remark 9. *Step 3 is so complicated in definition because of loops. We need to replace* Z *as start/end vertex of outgoing/incoming edge* a, *and also choose correctly only one of copies of* Z *in case* a *is a loop .*

---

[4]There are four possibilities: either $\text{start}(a) = \text{start}(b)$, either $\text{end}(a) = \text{start}(b)$, either $\text{end}(a) = \text{end}(b)$, either $\text{start}(a) = \text{end}(b)$.

Theorem 1. *After applying rectangle split operation to each vertex of initial graph* G *in any order we'll receive a graph* $G' = (V', E')$, *such that there exists one-to-one mapping from* (V', E') *to* (V_r, E_r) *and this mapping respects labeling.*

Remark 10. *Initial graph* G *is also correct ungluing of itself, with identical* f *and* g. *We'll iteratively modify* f *and* g *in each rectangle split operation:* $f(Z_i) := f(Z)$, $g(a_i) := g(a)$ *It's evident, that after each rectangle split ungluing conditions (from definition of correct ungluing) on* f *and* g *are still satisfied.*

Proof. Let's fix some order of rectangle split operations. While applying rectangle split operations to each vertex except Z, we can somehow modify P, but the set $P_v(Z)$ stays constant (because vertex-edge paired info, obtained from removed edge e can be still obtained from an edge e_i for some i). By construction of rectangle graph, there is a one-one mapping from C_i(and so from Z_i) to set of all blue-connected components, having Z as first vertex in label. Let's consider union of this mappings for all vertices $Z \in V$.

Consider any edge $a'_{ij} = (Z_i, Y_j)$ with label a. Without losing of generality we can assume that vertex Z of original graph G split before Y. While splitting Y, we have a precursor $((a_i, b), t)$ (that remapped into (a_{ij}, b, t)) of vertex-edge pair info's $(Y_j, b, t - l_{a_i}) \in D_j$, such that $end(a_i) = Y$ where $P_v(Y) = \bigcup D_j$. While splitting Z, we have a precursor $((a, b), t)$ of vertex-edge pair info's $(Z, b, t) \in C_i$, such that $start(a) = Z$. The edge pair info $((a, b), t)$ belongs to C_i and D_j, this means that edge pair info $((a, b), t)$ produce rectangle such that red edges of this rectangle connect two blue edges components that mapped into C_i and D_j, and this edges labeled by a. So, we shown that for each edge in our graph G' there exist an edge in G_{rb} which respects one-one mapping on vertices and has the same label.

Let's take red edge from G_{rb} with label a. It connects two blue-connected components. These components are mapped to C_i and D_j from G'. Without less of generality, Z was splitted before Y. So, by construction of G', there exists an edge $b \in E$: such that $(Z, b, x) \in C_i$ and precursor of this vertex-edge pair is $((a, b), x)$. There is a vertex $Z_i \in G'$ mapped to C_i in description of rectangle split operation. In step 4 of rectangle split operation, $((a, b), x)$ will be transformed into $((a_i, b), x)$. Then, consider the split of vertex Y. By construction, there exists $D_j : ((Y, b), x - l_a) \in D_j$ for $((Y, b), x - l_a)$ obtained from $((a_i, b), x)$. After splitting Y, a_i would be transformed into a_{ij}, where $end(a_{ij}) = Y_j$. So, we proved that after this two splits there'ld appear an edge a_{ij} connecting Z_i and Y_j, and no other splits wouldn't affect this edge. $\square$

Theorem 2. *In terms of Son's paper, result of his gluing procedure and contracting blue edges is a honest ungluing of initial graph* G.

sketch. Each rectangle split operation for honest ungluing of G produce honest ungluing of G, so if we apply them consequently, we'll receive a honest ungluing of G. $\square$

Theorem 3. *In terms of Son's paper, result of his gluing procedure and contracting blue edges has measure of complexity, which is less or equal to measure of complexity of initial graph* G.

Each rectangle split operation doesn't increase incoming-outgoing set(due to function g) of G, so their combination's incoming-outgoing set is also a subset of incoming-outgoing set of G.

5 Our ideas

5.1 Bad cases for rectangle graph

Rectangle graph is a powerful tool, but it's simple to construct graph that can be resolved manually, but stays unchanged after applying of rectangle graph techniques: Lets consider graph G with vertex set $V = A, B, C, E, F, G, H$ and edges

$$\{(A, B), (B, C), (C, E), (E, F), (F, G), (G, A), (C, H), (H, F)\}$$

. Pair information is set of all edge pairs with distance(on some subpath of G) 2 i.e. (A, B) and (C, E). Both red and blue rectangle graph are isomorphic to G but after using of slightly modified split operation for this graph we can receive a correct ungluing, consisting of two cycles: $(A \to B_1 \to C_1 \to E \to F_1 \to G_1 \to A)$ and $(A \to B_2 \to C_2 \to H \to F_2 \to G_2 \to A)$.

5.2 Rectangle split modifications

First approach is to modify adjacency definition for vertex-edge pairs.

Definition 20. *Two vertex-edge pairs* (Z, a, x) *and* (Z, b, y) *are called* adjacent *if* $\text{start}(a) = \text{end}(b)$ *or* $\text{start}(b) = \text{end}(a)$ *and distances from* Z *to common vertices of* a *and* b, *obtained from* (Z, a, x) *and from* (Z, b, y), *are same.*

This definition is slightly different from previous adjacency definition. Using this definition we can split vertex V by vertex-edge information even if there two edges paired with V has common start or end vertex.

In case of inexact insert length we have to somehow weaken the condition on distances. Also, for highly nonuniform coverage, the request of sharing vertex for adjacent edges should be weakened:

Definition 21. *Two vertex-edge pairs* (Z, a, x) *and* (Z, b, y) *are called* $l-\varepsilon$ adjacent *if there exists path* P *from* a *to* b: $|P| < l$ *and* $\|P| - |y - x\| < \varepsilon$.

5.3 Usage of split operations

After performing rectangle split for all vertices of initial graph we can recalculate pair information and do split for new vertices. It can be useful if after splitting of all vertices from initial graph, some connected components of pair infos break into unconnected parts. Also, for some vertices it is possible to split them by using paired information in other direction - consider *edge-vertex info* (a, V, x) and look on adjacency(defined in same way as for vertex-edge infos) for such objects. This splitting is equivalent (in terms of rectangle graph) to contracting red edge.

5.4 Incorporation of mate-read transformation (still under discussion!!!)

Let's consider original rectangle split operation.

After applying rectangle split operation to every vertex of initial graph we'll receive graph G' and a "hybrid" paired info- set of edge pairs, where first element is an edge from G', and second- an edge from G.

Let's translate it to terms of graph G':

Definition 22. *As* paired multiedge info *we'll call* $((e_1, S_{e_2}), x)$, $e_1 \in E', S_{e_2} \subset E'$, *saying that* $\exists e_{2i} \in S_{e_2}$ *such that distance between* e_1 *and* S_{e_2} *is equal to* x.

If there is no path of length x between e_1 and e_{2i}, it means that edge e_1 and e_{2i} are not in distance x, so in this case e_{2i} can be removed from S_{e_2} !

We hope that such 'non-standart' mate-pair transformation can be useful in repeat resolving.

So, now we forget about G - every edge below is edge from G'. Also, we can remove all one-one vertices, merging it's incoming and outgoing edges and corresponging paired multiedge info

Then we can define vertex-multiedge pair info in similar way and define adjacency for vertex-mutiedge pair info as something like "There exists an edge e_1 from first edge-set and e_2 from second edge- set such that they share a vertex and distances to this vertex provided from both pair infos are equal".

Now we can do an analogue of rectangle split operation with vertex-multiedge pair info instead of vertex-edge pair info and receive a graph G''

Again, it's easy to prove that it is honest ungluing.

Then again we can do the same with graph G'', and repeat this until nothing is changed.

Remark 11. *This method seems to be resistant to all modifications of original rectangle split operation excluding "back" splitting.*
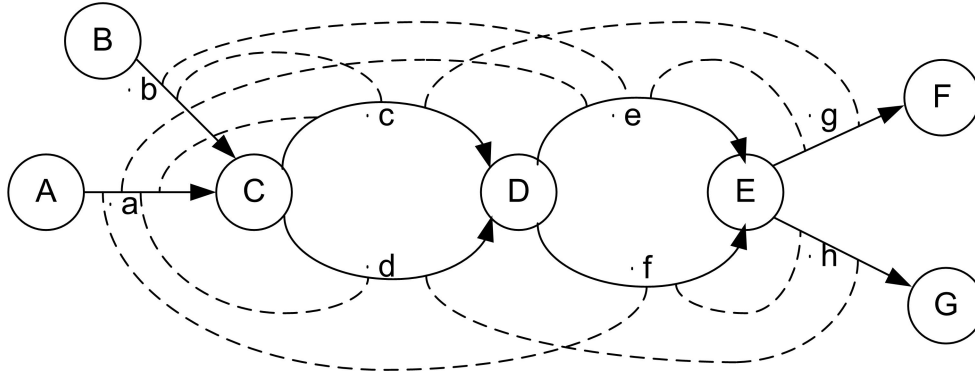
Figure 1: Graph example

6 Details about real implementation

We have to ignore low confirmed information about distance between edges to cope with chimeric pairs. Also, it may turn out that after out resolving we'll need to do tip clipping again because of erroneous connections.

7 Connected ideas

7.1 Insert-length distribution specifying

We can use mate-pairs aligning the same edge to specify insert-length distribution.

7.2 Complicated repeat resolving

Next step is to resolve repeats using distances distributions on neighbouring edges (to divide sum of two close normal distributions with one). Also, in case of big coverage we can try to correct distance information. For example, we can obtain information about distances to vertex from distances to it's incoming and outgoing edges. This two series of distances should be same. If they aren't, we can find minimal correction to get same distances and apply it to both of series.

7.3 Extending the model

In this paper we used only paired information. It seems that coverage can also be introduced to our model.

8 Example

Let's consider the graph on fig. 1. Dotted lines connect edges on distance d. Let's consider vertex E. The vertex-edge paired information obtained from neighboring edges

is $\{(E, g, 0), (E, h, 0)\}$. In terms of rectangle split operation this is one connected component, but in terms of our slightly modified split operation there are two components. We split E into E_g and E_h, and edges remaps into this vertices as follow: $e \to (D, E_g)$, $g \to (E_g, F)$, $f \to (D, E_h)$, $h \to (E_h, G)$.

Now let's consider vertex C. We obtain next set of vertex-edge pair infos:

$$\{(C, c, 0), (C, e, l_c), (C, g, l_c + l_e), (C, d, 0), (C, f, l_d), (C, h, l_d + l_f)\}.$$

If lengths l_c and l_d are equal then all pair infos above belong to one component. In this case we can not split anything. Furthermore, there are no reasons to do this split if we know only original graph and only this infos.

Let's see the vertex D. Infos are $\{(D, g, l_e), (D, h, l_f)\}$. By our rule this set can be splitted into two components. For first component $\{(D, g, l_e)\}$ we create vertex D_g, and have two precursors: $(c, g, *)$, and $(e, g, *)$. So we remap edge c into (C, D_g), and e into (D_g, E_g). For second component $\{(D, h, l_f)\}$ we have vertex D_h and remap d into (C, D_h) and f into (D_h, E_h).

Now we can return to vertex C. Set of vertex-edge pair infos is the same

$$\{(C, c, 0), (C, e, l_c), (C, g, l_c + l_e), (C, d, 0), (C, f, l_d), (C, h, l_d + l_f)\}.$$

But now edge c and edge f no more adjacent, because c remapped into vertex D_g, but f remapped into vertex D_h. The same happens with edges d and e. Now we have TWO component $\{(C, c, 0), (C, e, l_c), (C, g, l_c+l_e)\}$, and $\{(C, d, 0), (C, f, l_d), (C set of information, h, l_d+l_f)\}$. For first component we create vertex C_g. We obtain next remapping of edges: $a \to (A, C_g)$, $b \to (B, C_g)$, $c \to (C_g, D_g)$. For second component we create vertex C_h and obtain remapping $a \to (A, C_f)$, $d \to (C_f, D_f)$. For both this component we remap edge a. And now we have two edge marked by a.

So we split all possible vertices.