

Multiple-source shortest paths with unit weights in embedded graphs

Abstract

We describe a new algorithm that computes multiple-source shortest paths from vertices in a given boundary face to all other vertices in an embedded graph with unit weight edges.

1 Introduction

Recently, Eisenstat and Klein [2] introduced a new algorithm for computing multiple source shortest path problem for a planar graphs in linear time. Our paper attempts to generalize this idea to embedded graphs. In their paper, they maintain so-called leafmost shortest tree to get around an issue of ambiguous shortest paths between a pair of vertices. There is no direct way to generalize a leafmost tree computing process to an embedded graphs, since leafmost tree is not well defined data structure in an embedded graphs. In the following section we introduce terms that alleviate the process of ambiguous pivoting.

Introduce notations, homology, and dual graphs.

We can formally define multiple-source shortest paths problem as follows:

Given. Let G be a directed graph (V, \vec{E}) , embedded on a surface with genus g . All edge weights are unit and $|V| = n$. Using Euler's theorem, we can derive $|E| = n + 2g - 1$.

Find. Consider boundary face f of G . $\forall v \in f$, find a shortest path to $\forall u \in V$.

2 Holy Tree

Let T be a breadth first search(BFS) tree of G , and C^* be a BFS co-tree in $(G/T)^*$. Then there is exactly $2g$ leftover edges $L = \{e_1, e_2, \dots, e_{2g}\}$, according to the Euler's formula.

There exists a unique cycle λ_i in $C^* \cup e_i$, and $(\lambda_1, \lambda_2, \dots, \lambda_{2g}) = \Lambda$ defining homology basis. We define homological signature of an edge as follows:

$$[e]_i = \begin{cases} 1 & , \text{if } e \in \lambda_i \\ -1 & , \text{if } rev(e) \in \lambda_i \\ 0 & , \text{o/w} \end{cases}$$

Furthermore, we define leafmost term α recursively as follows:

$$\alpha(e^*) = \begin{cases} 1 & , \text{if } rev(e^*) \text{ is a leaf dart in } C^* \\ \sum_{tail(e'^*)=head(e^*)} \alpha(e') & , \text{o/w} \end{cases}$$

We can extend above definition with $\alpha(e) = \alpha(e^*)$ and $\alpha(e)^* = -\alpha(rev(e^*))$.

This leafmost term definition has been first introduced by Cunningham in ???.

Let $\tilde{w}(e) = (1, [\vec{e}], \alpha(e))$ be new weight vector for each edge in G . We refer to each component this weight vector as length, homology, and leafmost terms respectively.

Definition. An edge e_1 is holier than e_2 , if $\tilde{w}(e_1) < \tilde{w}(e_2)$ in lexicographic comparison. Therefore, we can define holiness of any $S \subset G$ as follows:

$$H(S) = \sum_{e \in S} \tilde{w}(e)$$

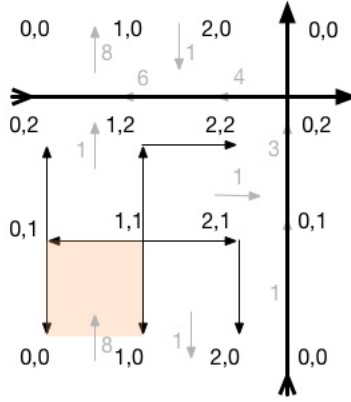


Figure 1: An example of holy tree in graph with genus 1.

Note here that computing holy tree rooted at s immediately gives us single source shortest path tree rooted at s in our case.

We show an example figure for holy tree computation for a grid graph embedded in genus 1 surface. Notice that boundary edges are replicated for nice visual. Bold thick arrows represent homology cycles, grey arrows correspond to leafmost terms. Holy tree is rooted at $(1,1)$.

3 Need of holiness

We provide a simple explanation on why it is necessary to introduce holiness to cope with ambiguity.

1. Leafmost term definition is not enough. This is because leftover $2g$ edges could still cause ambiguity when computing shortest paths.
2. By introducing homology terms, we avoid this ambiguity.

4 Leftmost tree vs Holy tree

1. Explain our initial attempts on leafmost tree maintaining to show it is NOT sufficient for dealing with non-unique shortest paths.

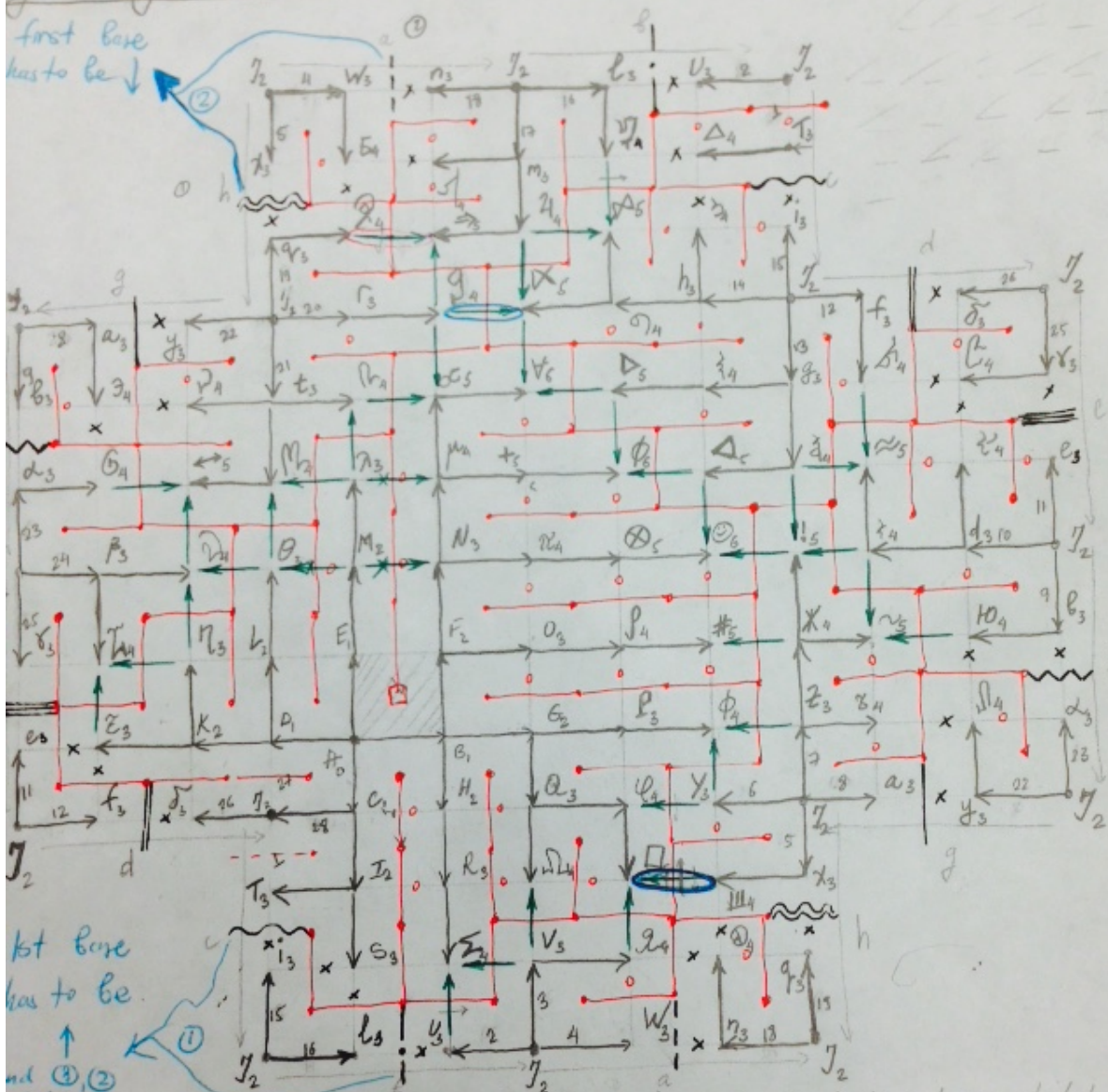
Klein [3] uses a notion of leftmost tree to resolve ambiguity when computing MSSP. Our initial hope was that even in higher genus surfaces leftmost tree and holy tree would be equal. However, on genus $g = 4$ surface, we show that no choice of homology and leafmost terms can result in equal holy and leftmost trees. Consider below picture,

- Leftmost tree is rooted at A and noted by solid black darts.
- There are exactly 8 homology cycles, each specifically denoted with special darts crossing the boundaries.
- Leafmost term building tree is denoted with red darts.

The observation on the figure is 2 darts highlighted in blue will be tense, regardless of the homology and leafmost choice configurations.

genus $g = 4$ surface:

first base
has to be \downarrow



1st base
has to be

order
③, ②

$$1 \downarrow 2 \downarrow \text{sha}(\text{②} \rightarrow \text{③}) = (\text{③}, \text{②}, \text{②}, \dots) + (\text{③}, \text{③}, \text{③}, \dots)$$

$$\text{③} \text{ place } (\text{③} \rightarrow \text{③}) = 5(0, -1, 0, 1, 2, \dots)$$

$$1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100$$

5 Initial holy tree

Holiest tree is a spanning tree with minimal holiness. We build holiest tree rooted at r , using slight tweak in the Bellman-Ford algorithm for finding shortest path tree rooted at r .

BuildHoliestTree(G, \tilde{w}, r):

```

Set  $dist[r] \leftarrow (0, [\vec{0}], 0)$ 
 $pred(r) \leftarrow \text{NULL}$ 
for all  $v : v \neq r$ 
     $dist[v] \leftarrow (\infty, [\vec{\infty}], \infty)$ 
     $pred(v) \leftarrow \text{NULL}$ 
put  $r$  into queue
while queue is not empty:
    Let  $u \leftarrow$  dequeue item
    for all  $u \rightarrow v$ 
        if  $v$  is not marked
            mark  $v$  and put in the queue
        if isTense( $u \rightarrow v$ )
            relax( $u \rightarrow v$ )

```

isTense($u \rightarrow v$):

return $dist[u] + \tilde{w}(u \rightarrow v) < dist[v]$

relax($u \rightarrow v$):

$dist[v] \leftarrow dist[u] + \tilde{w}(u \rightarrow v)$
 $pred[v] \leftarrow u$

Observation. Each vertex will be added once to the queue.

Corollary. Each edge will be relaxed at most once.

Lemma 2.1. If there is no tense edge in G , then for each $v : r \rightarrow \dots \rightarrow pred(pred(v)) \rightarrow pred(v) \rightarrow v$ is the holiest path from r to v .

Proof: Let's prove it by induction on $dist[v].length$ distance from the root r .

Base. $dist[v].length = 0$, then $v = r$, so the claim holds trivially.

Induction Step. Suppose the claim is true for all vertex $v \in V$ such that $dist[v].length < d$ for some d . Consider vertex v such that $dist[v].length = d$. By induction hypothesis, all vertices with $dist[u].length = d - 1$ have holiest path correctly updated. By definition, $dist[v] = \min_{u \rightarrow v} \{dist[u] + \tilde{w}(u \rightarrow v)\}$, here $dist[u].length = d - 1$. By Induction hypothesis, $dist[u]$ is not tense and can construct holiest path to u , so if there is no tense edge in G then $dist[v] = \min\{dist[u] + \tilde{w}(u \rightarrow v)\}$ holds.

□

Corollary. The algorithm will produce holiest tree rooted at r in $O(n + g)$ time.

6 Moving Along an Edge

Let T_u be a current holiest tree, $u \rightarrow v$ be an edge that we are trying to move along. So at the end of the process we would like to obtain T_v , the holiest tree rooted at v .

We follow Cabello, Chambers, and Erickson's [1] method to move across the edge, by bisecting the given edge and inserting new source s which is connected to both u and v . At the start of the process, $s == u$ and continuously move to v , and when $s == v$, we would have our $T_s = T_v$.

Initial attempt:

Let $(1, \vec{h}, \alpha) = w(u \rightarrow v)$. We treat λ as a parameter with satisfying following equations:

$$w_u(s \rightarrow u) = (0, [\vec{0}], 0), \quad w_u(s \rightarrow v) = (1, \vec{h}, \alpha) \quad (1)$$

$$w_v(s \rightarrow u) = (1, -[\vec{h}], -\alpha), \quad w_v(s \rightarrow v) = (0, \vec{0}, 0) \quad (2)$$

The natural definition would be as follows, but it does not satisfy our constraints (1) – (2).

$$w_\lambda(s \rightarrow u) = (\lambda, [\vec{0}], 0) \quad (3)$$

$$w_\lambda(s \rightarrow v) = (1 - \lambda, \vec{h}, \alpha) \quad (4)$$

Therefore, we modify our parametric definition by decreasing distance to u by $w(u \rightarrow v) = (0, -[\vec{h}], -\alpha)$. This change allows us to define

$$w_\lambda(s \rightarrow u) = (0, -[\vec{h}], -\alpha) + \lambda * (1, [\vec{h}], \alpha) \quad (5)$$

$$w_\lambda(s \rightarrow v) = (1, [\vec{h}], \alpha) + \lambda * (1, [\vec{h}], \alpha) \quad (6)$$

Since we decrease distance to u at the start, this could potentially introduce pivots. Suppose, for instance, x be descendant of u and y be of v in T_s , then:

$$slack(x \rightarrow y) = dist(x) + w(x \rightarrow y) - dist(y) = \quad (7)$$

$$= dist_0(x) + (0, -[\vec{h}], -\alpha) + w(x \rightarrow y) - dist_0(y) = \quad (8)$$

$$= slack_0(x \rightarrow y) + (0, -[\vec{h}], -\alpha) \Rightarrow \quad (9)$$

$$slack(x \rightarrow y) < 0, \text{ if } slack_0(x \rightarrow y) < (0, [\vec{h}], \alpha) \quad (10)$$

Therefore, this could be potentially problematic. Furthermore, **What if $u \rightarrow v$ has non-trivial homology or leafmost term?**

We need to take into account that $u \rightarrow v$ has non-trivial homology and leafmost terms.

1. First attempt of not maintain holy tree at all times.
2. Second attempt of continuously moving by changing homology and leafmost terms, also reducing destination distance initially.
3. Necessity of resolving homology and leafmost values for an edge that source is moving.

$$w_0(s \rightarrow u) = (0, [\vec{0}], 0) \quad (11)$$

$$w_0(s \rightarrow v) = (1, [\vec{h}], \alpha) = w(u \rightarrow v) \quad (12)$$

Observe that this condition implies $s = u$, therefore $T_s = T_u$. We reduce distances to u and v as follows:

$$w_\epsilon(s \rightarrow u) = (0, -[\vec{h}], -\alpha) \quad (13)$$

$$w_\epsilon(s \rightarrow v) = (1, [\vec{0}], 0) \quad (14)$$

Since we reduced distance to all vertices in the graph equally, the process does not introduce any pivots. Then we define a parametric weights as follows:

$$w_\lambda(s \rightarrow u) = (0, -[\vec{h}], -\alpha) - \lambda \quad (15)$$

$$w_\lambda(s \rightarrow v) = (1, [\vec{0}], 0) + \lambda \quad (16)$$

Every other dart $x \rightarrow y$ has constant parametric weight $w_\lambda(x \rightarrow y) = w(x \rightarrow y)$. We then maintain the holy tree T_λ rooted at s , with respect to the weight function w_λ , as λ increases continuously from 0 to $(1, [\vec{0}], 0)$. When $\lambda = w(u \rightarrow v)$, $T_\lambda = T_v$.

In the following algorithm, **pred** defines holy tree rooted at u , and **dist** is corresponding distance to each vertex in the graph.

```

MoveAlongEdge( $G, u \rightarrow v, dist, pred$ ):
  Add new vertex  $s$ 
   $pred[u], pred[v] \leftarrow s$ 
   $\lambda \leftarrow 0$ 

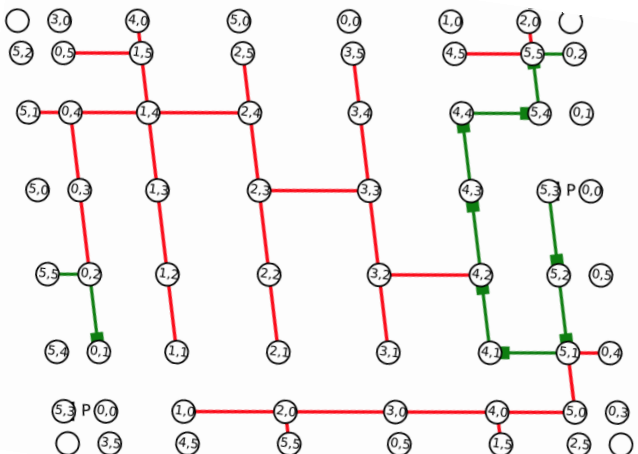
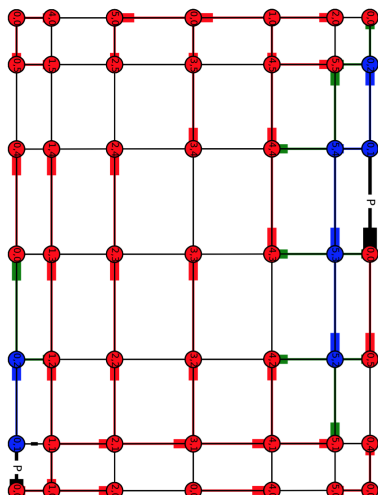
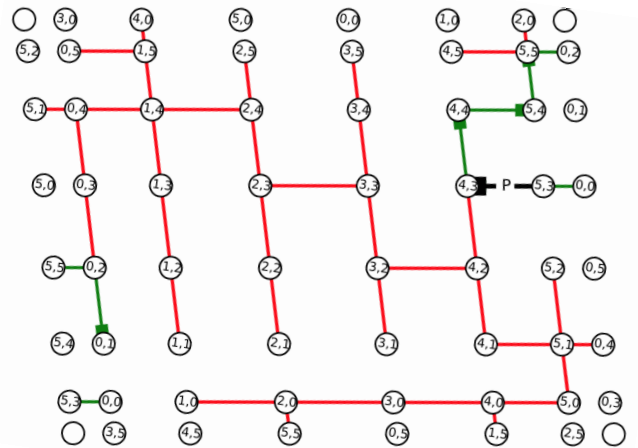
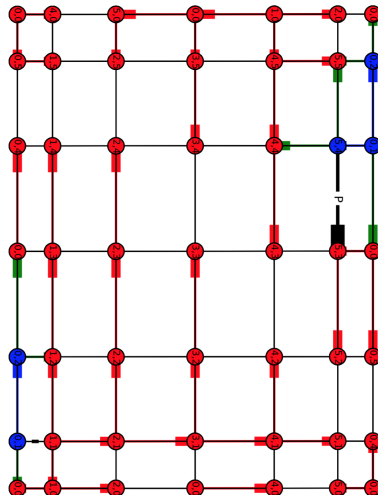
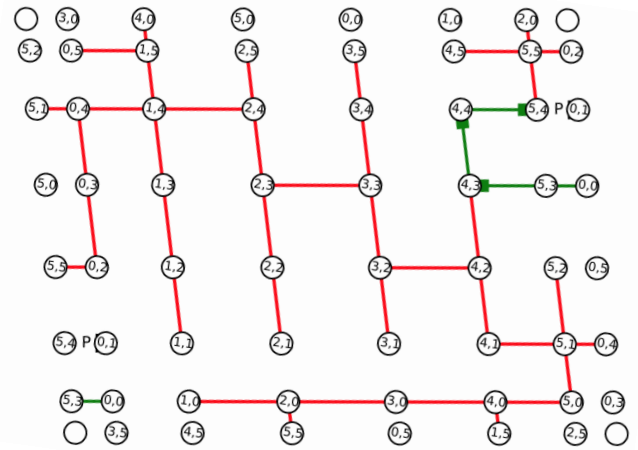
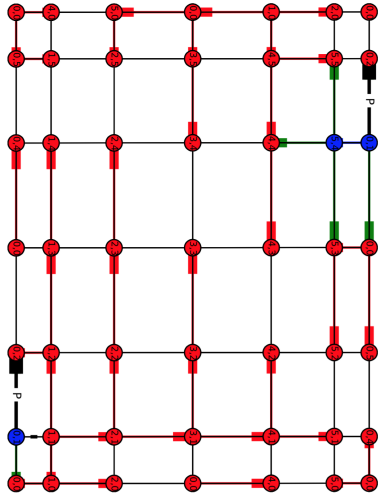
   $w(s \rightarrow u) \leftarrow (0, -[w(s \rightarrow u)], -\alpha(w(s \rightarrow u)))$ 
  AddSubtree( $(0, -[w(s \rightarrow u)], -\alpha(w(s \rightarrow u)))$ ,  $u$ )

   $w(s \rightarrow v) \leftarrow (1, [\vec{0}], 0)$ 
  AddSubtree( $(0, -[w(s \rightarrow u)], -\alpha(w(s \rightarrow u)))$ ,  $v$ )

  while  $\lambda < (1, [\vec{0}], 0)$ :
    pivot  $\leftarrow$  FindNextPivot
    If pivot is non NULL AND  $(\lambda + slack(\mathbf{pivot})/2) < (1, [\vec{0}], 0)$ 
      Pivot(pivot)
       $\lambda \leftarrow \lambda + slack(\mathbf{pivot})/2$ 
    else
       $\delta = (1, [\vec{0}], 0) - \lambda$ 
      AddSubtree( $\delta$ ,  $u$ )
      AddSubtree( $-\delta$ ,  $v$ )
       $\lambda \leftarrow \lambda + \delta$ 

```

Below, we show an example of moving along an edge process on genus $g = 2$ grid. On a left side, we see primal holy tree H with nodes increasing in distance are red, decreasing are blue, and next pivot is denoted as P . On right side, we see the dual graph $(G/T)^*$ and active darts are noted with green color.



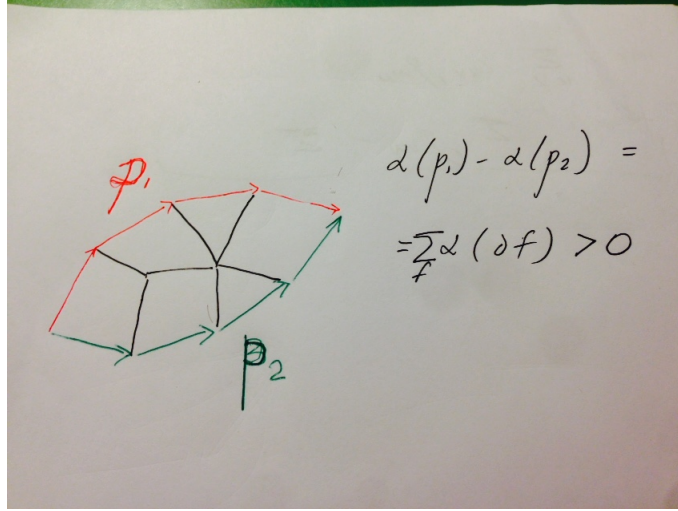
7 Bounding number of pivots

1. Describe how Eisenstat Klein clock lemma works and why it is not easily generalizable in higher genus graphs.

We introduce a clocking lemma to prove that each edge is involved in pivoting process at most $O(g)$ times.

Lemma 4.1. Let v_0 be a first vertex in our given face f , and v_i be the source right after i^{th} pivot. Consider a vertex y . We denote holiest path from v_i to y as P_i^y . Then P_i^y and P_j^y are non-crossing for all i, j .

Proof: Consider the following picture:



By the definition of α :

$$\alpha(p_1) - \alpha(p_2) = \sum_f \alpha(\partial f) > 0$$

This will ensure that any two paths p_1, p_2 , whose $w(p_1) = w(p_2)$ and $[p_1]_\Lambda = [p_2]_\Lambda$, has $\alpha(p_1) \neq \alpha(p_2)$ □

Lemma 4.2. As source vertex s moves around the given face f , any dart d has exactly $2g$ continuous clock state:

$$d \in T \tag{17}$$

$$d^* \in (G/T)^* \tag{18}$$

$$\text{rev}(d) \in T \tag{19}$$

$$\text{rev}(d^*) \in (G/T)^* \tag{20}$$

Proof: Here is the proof. □

Theorem 4.1. Total running time of MSSP is $O(gn)$.

Proof: Building initial holy tree takes $O(n + g)$ time. The process of moving around the face and pivoting takes $O(gn)$ as each dart enters and replaces $O(g)$ times. □

8 Finding pivot quickly

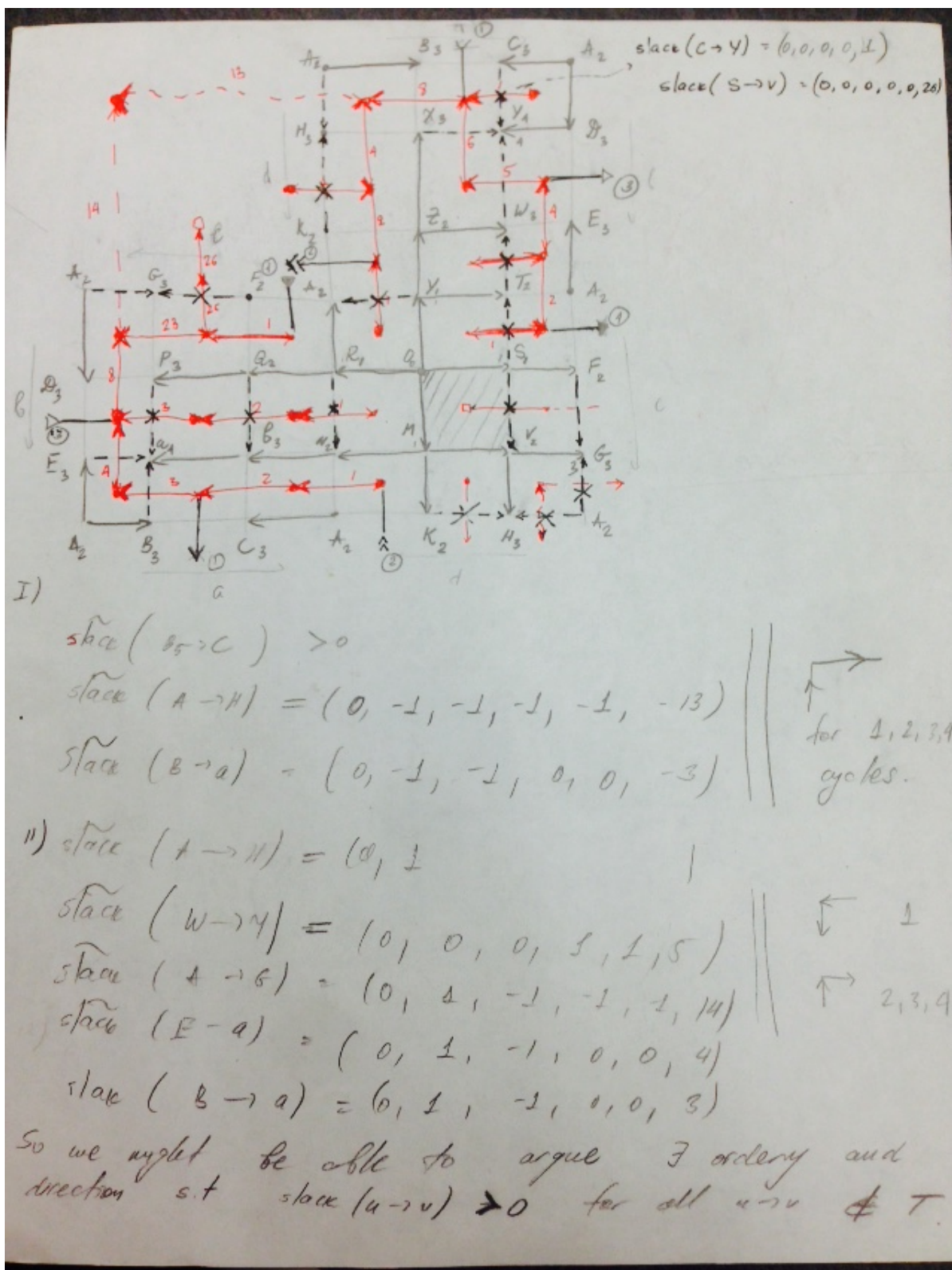
- What data structure do we maintain in the G^* ? Finding shortest path in network can also be understood as a Linear Programming problem as follows:
- How do we find next pivot quickly using above structure?

9 Analysis

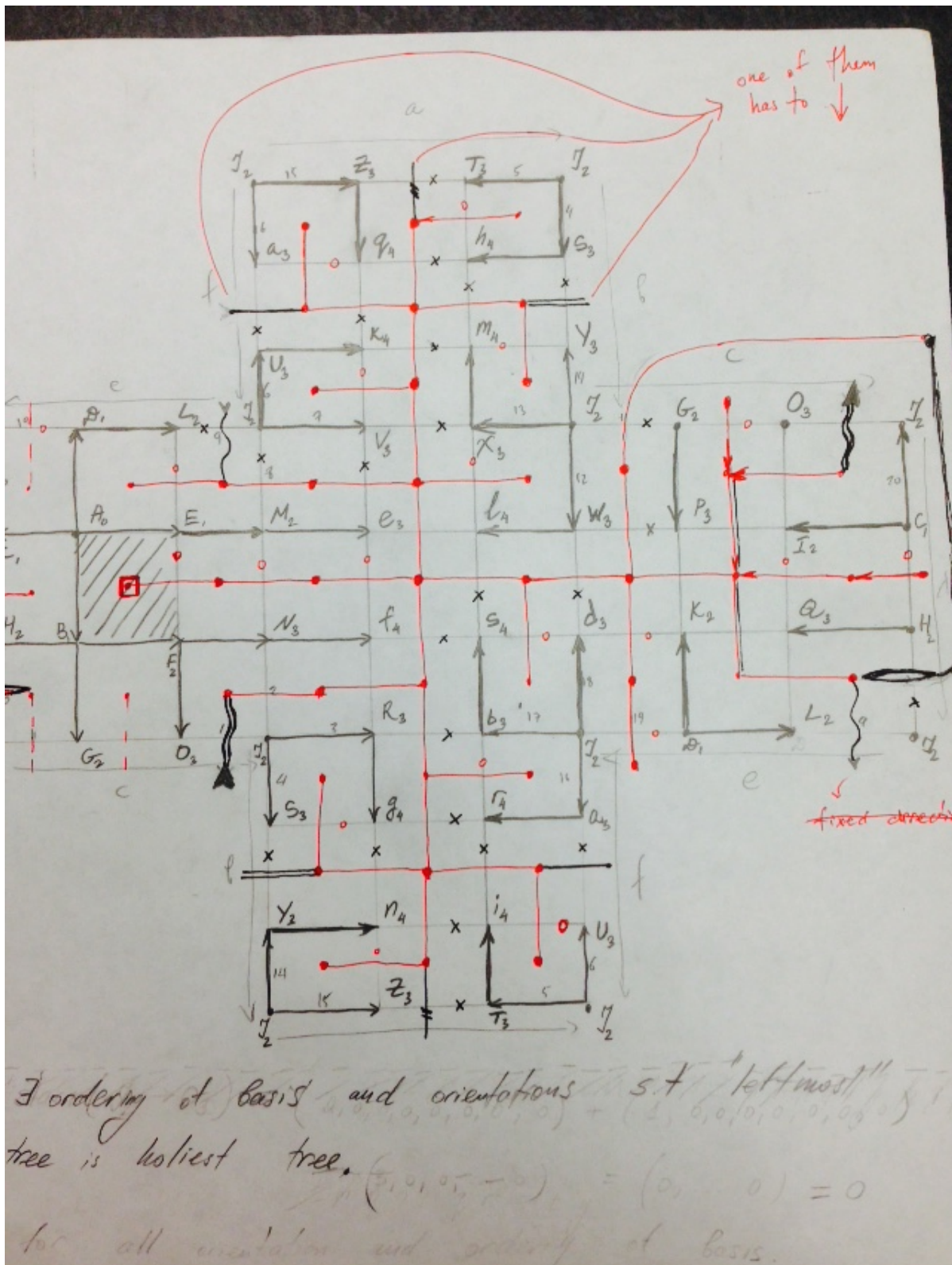
- Building initial tree
- Pivoting
- Number of times each edge is pivoted
- Overall running time

10 Working on examples:

On genus $g = 2$ surface:



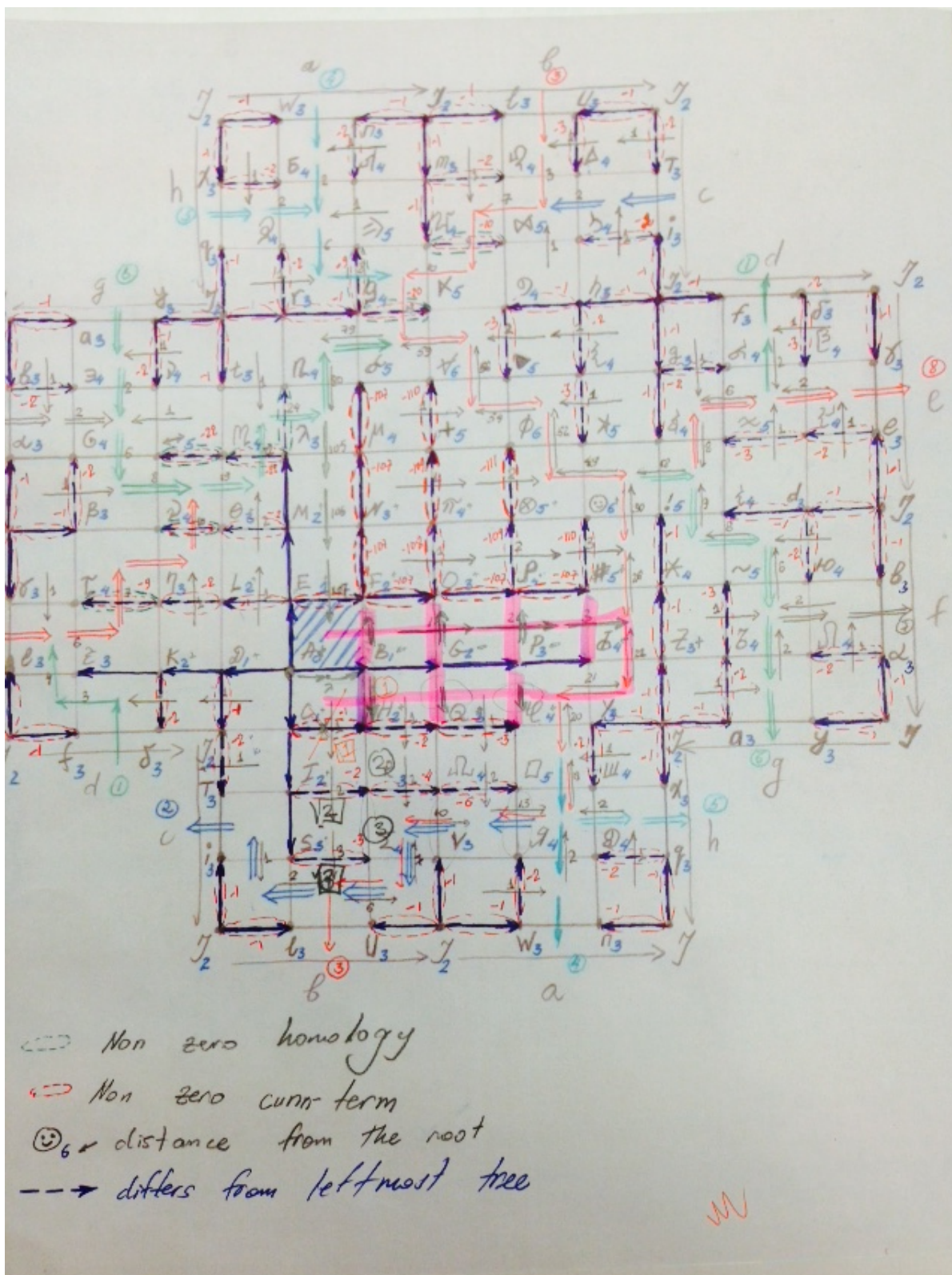
On genus $g = 3$ surface:



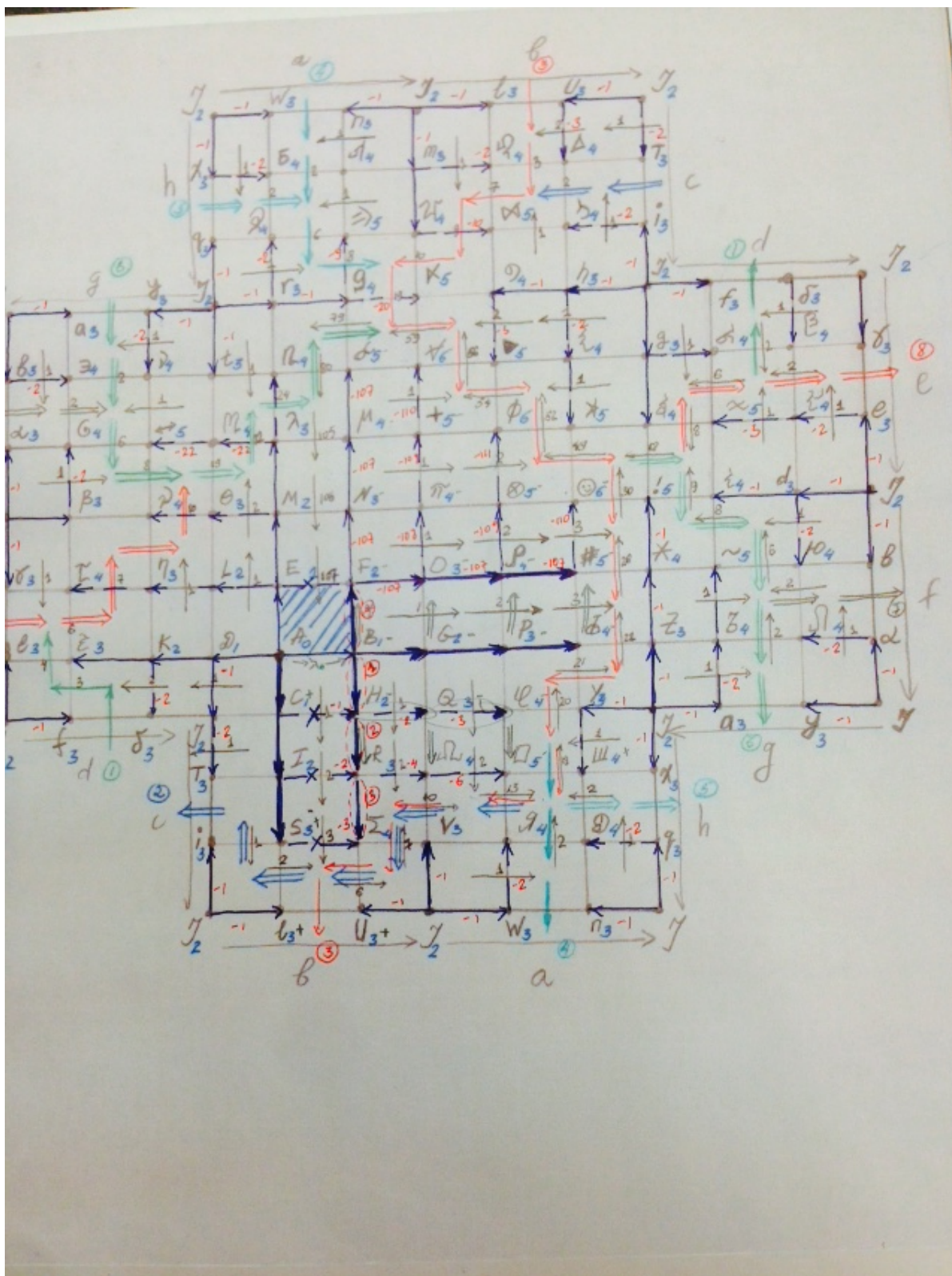
~~fixed direction~~

For ordering of basis and orientations s.t. "leftmost" tree is holiest tree. $(\pm, 0, 0, 0, 0, 0, 0, 0) = (0, \dots, 0) = 0$ for all orientation and ordering of basis.

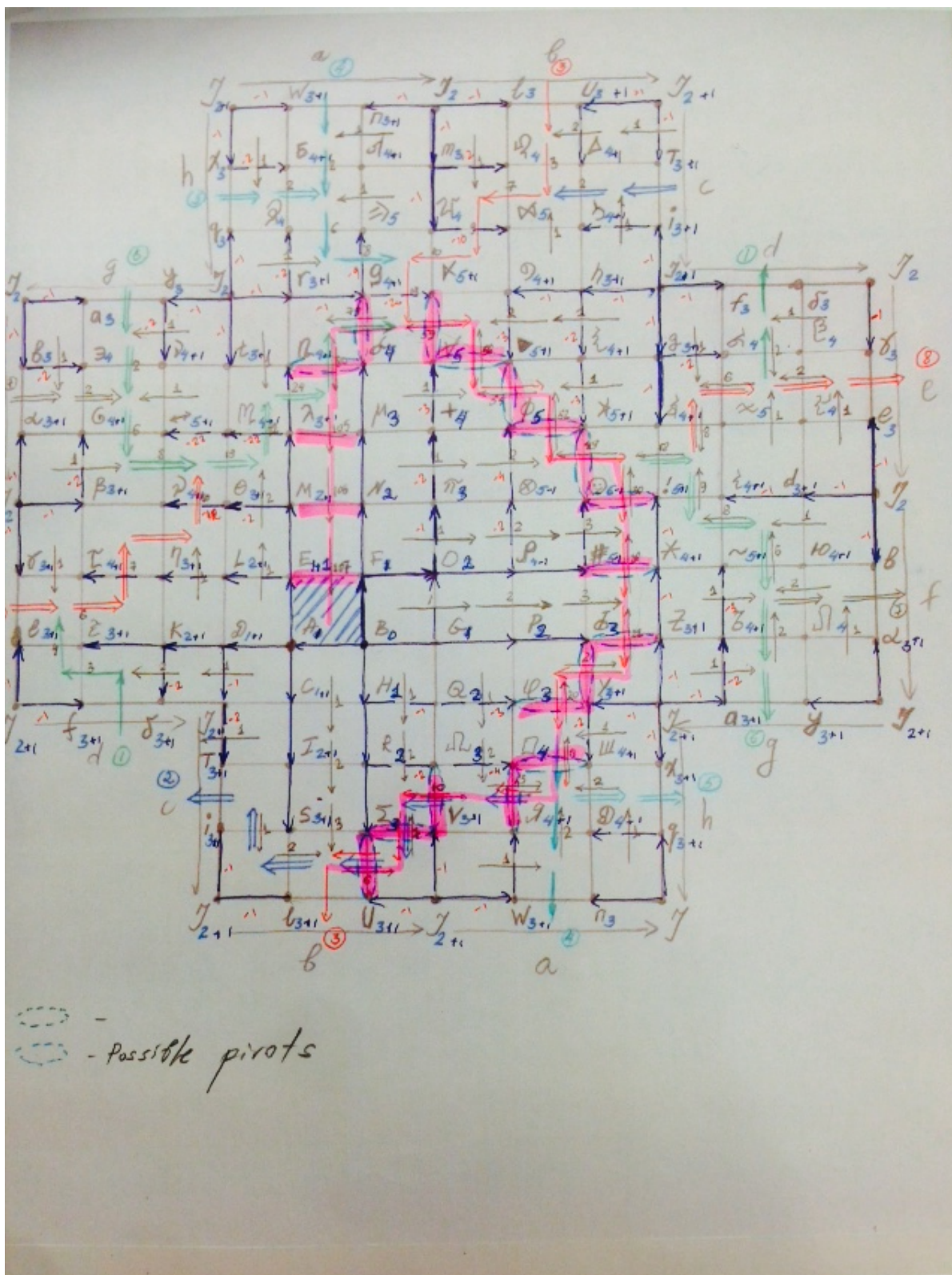
On genus $g = 4$ surface with initial pivot:



On genus $g = 4$ surface with initial pivot:



On genus $g = 5$ surface with initial pivot:



References

- [1] Sergio Cabello, Erin W Chambers, and Jeff Erickson. “Multiple-source shortest paths in embedded graphs”. In: *SIAM Journal on Computing* 42.4 (2013), pp. 1542–1571.
- [2] David Eisenstat and Philip N Klein. “Linear-time algorithms for max flow and multiple-source shortest paths in unit-weight planar graphs”. In: *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*. ACM. 2013, pp. 735–744.
- [3] Philip N Klein. “Multiple-source shortest paths in planar graphs”. In: *SODA*. Vol. 5. 2005, pp. 146–155.