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 face to all other vertices in an embedded graph with unit weight edges.

1 Introduction

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

Given. *Let G be a directed graph $(V, \vec{E})$ with unit edge weights, embedded on a surface with genus g.*

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

Klein [15] described first algorithm to solve MSSP problem in planar graphs in $O(nlogn)$-time. Later, Cabello, Chamber, and Erickson [5] described an alternate method and generalized it to graphs embedded in higher genus surfaces. Recently, Eisenstat and Klein [9] introduced a new algorithm to compute MSSP in planar graphs in linear time when edge weights are unit. Our paper attempts to generalize this idea to graphs embedded in higher genus surfaces. In their paper, Eisenstat and Klein use so-called leafmost pivoting rule to deal with possibility more than one pivots at the given time. However, there is no direct way to generalize a leafmost pivoting rule in graphs embedded in higher genus surfaces In this paper, we describe ways to alleviate this difficulties and how to compute MSSP.

1.1 Definition and Notation

We adapt definitions and notations of surfaces, embedding, and duality from Cabello et al.'s paper [5]. Furthermore, we refer to graphs embedded on higher genus surfaces as "embedded graphs".

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 $L = \{e_1, e_2, \ldots, e_{2g}\}$, according to the Euler's formula.

There exists a unique cycle λ_i in $C^* \cup e_i$, and $(\lambda_1, \lambda_2, \ldots, \lambda_{2g}) = \Lambda$ 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 α term recursively as follows:

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

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

This α term definition was first introduced by Cunningham [8].

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 α terms respectively.

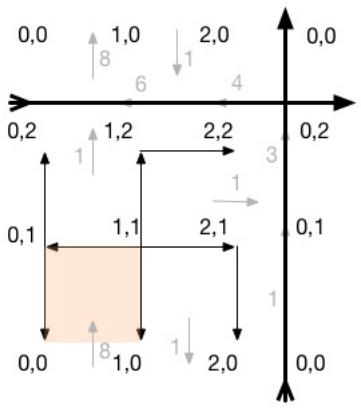


Figure 1: An example of holy tree in graph with genus 1.

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:*

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

Note here that computing holy tree rooted at s immediately gives us singe 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 α terms. Holy tree is rooted at $(1, 1)$.

2.1 Need of holiness

Lemma 2.1. *Let p_1 and p_2 be any two distinct paths between vertices x and y in an embedded graph G. Then $H(p_1) \neq H(p_2)$ holds.*

Proof: Suppose $H(p_1) = H(p_2)$ holds. Let γ be a cycle formed by concatanating p_1 and $rev(p_2)$.

$$H(p_1) = H(p_2) \Rightarrow H(p_1 + rev(p_1)) = 0 \Rightarrow H(\gamma) = 0 \Rightarrow \quad (1)$$
$$\Rightarrow [\gamma] = 0 \text{ and } \alpha(\gamma) = 0. \quad (2)$$

$[\gamma] = 0$ implies that γ is null-homologous, therefore a separating cycle in Σ. Suppose we get two separate connected components Σ_1 and Σ_2, which includes face f, if we split Σ with γ.

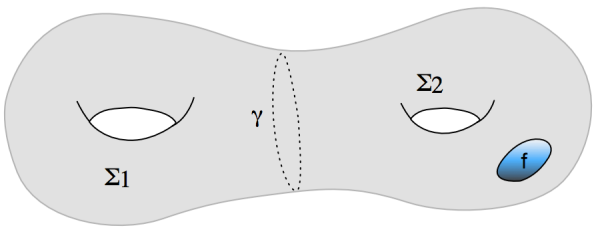


Figure 2: Separating cycle γ.

According to the definition of α term, $|\alpha(\gamma)|$ is exactly equal to number of faces in Σ_1. However, there is at least one face in Σ_1, so $|\alpha(\gamma)| > 0$, which contradicts to $|\alpha(\gamma)| = 0$ in (2). □

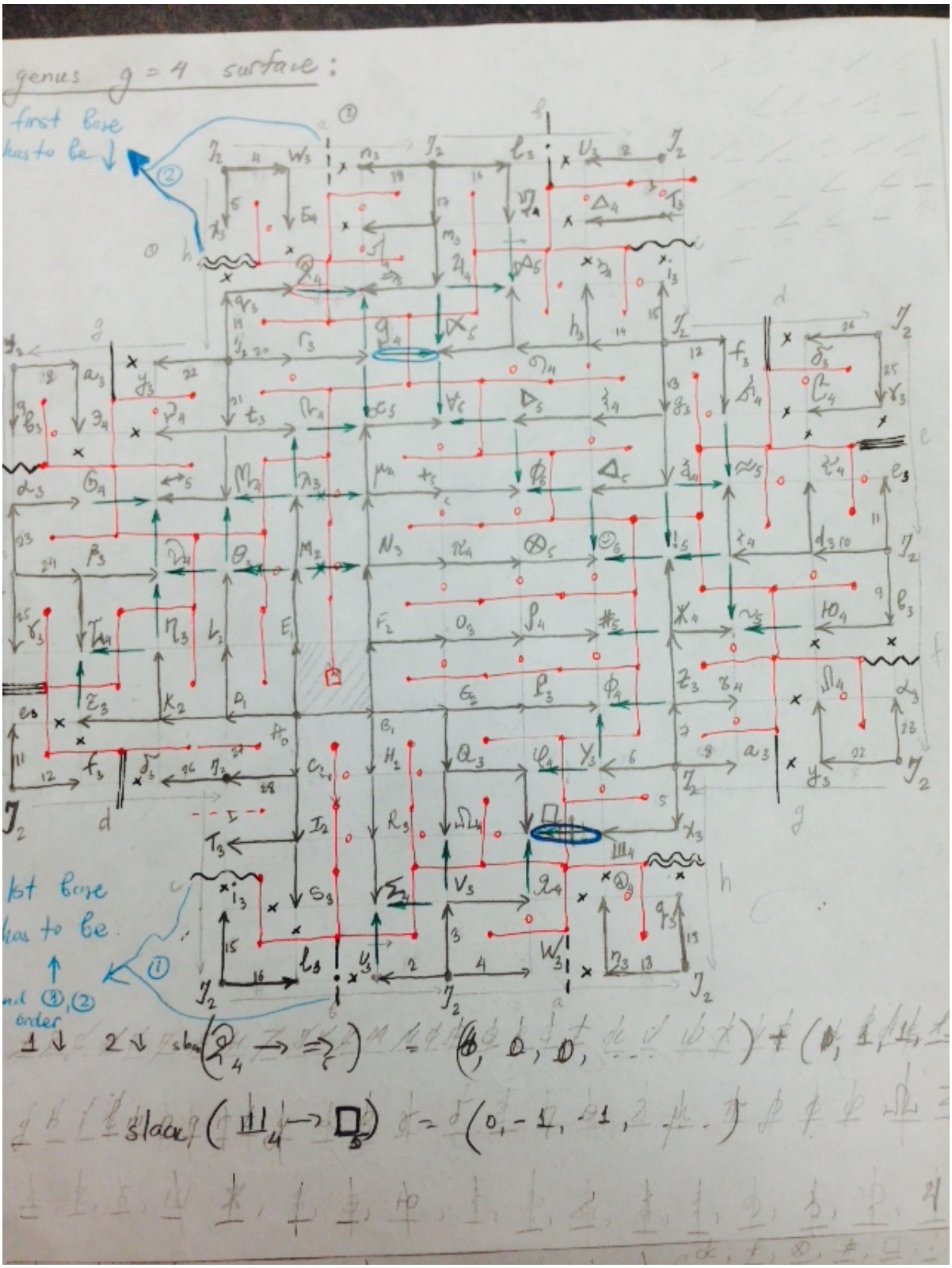
Corollary 2.1. *$\forall v \in f$, there is a unique holiest path from v to $\forall u \in V$.*

2.1.1 Leftmost tree vs Holy tree

Klein [15] used a notion of leftmost tree to resolve ambiguity when computing MSSP in their paper. 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 ordering and directions of homology and α terms can result in equal holy and leftmost trees. Consider below picture,

- Leftmost tree is rooted at A and denoted by solid black darts.
- There are exactly 8 homology cycles, each specifically denoted with special darts crossing the boundaries.
- α 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 α choice configurations.



2.2 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$(r) \leftarrow$ NULL
  for all $v : v \neq r$
    $dist[r] \leftarrow (\infty, [\vec{\infty}], \infty)$
    pred$(r) \leftarrow$ 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 $v : r \rightarrow \ldots \rightarrow pred(pred(v)) \rightarrow pred(v) \rightarrow v$ 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.*

2.3 Moving Along an Edge

We follow Cabello et al's [5] method to obtain shortest tree T_v from T_u. In our case, we are given holiest tree T_u and would like to obtain T_v, holiest tree rooted at v, by performing series of pivots.

We bisect the given edge uv and insert new source s, which is connected to both u and v. At the start of the process, $s == u$, and s move continuously from u 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:

$$w_u(s \rightarrow u) = (0, [\vec{0}], 0), \quad w_u(s \rightarrow v) = (1, \vec{h}, \alpha) \tag{3}$$

$$w_v(s \rightarrow u) = (1, -[\vec{h}], -\alpha), \quad w_v(s \rightarrow v) = (0, \vec{0}, 0) \tag{4}$$

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

$$w_\lambda(s \to u) = (\lambda, [\vec{0}], 0) \tag{5}$$
$$w_\lambda(s \to v) = (1 - \lambda, \vec{h}, \alpha) \tag{6}$$

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

$$w_\lambda(s \to u) = (0, -[\vec{h}], -\alpha) + \lambda * (1, [\vec{h}], \alpha) \tag{7}$$
$$w_\lambda(s \to v) = (1, [\vec{h}], \alpha) + \lambda * (1, [\vec{h}], \alpha) \tag{8}$$

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:

$$\begin{aligned}
&slack(x \to y) = dist(x) + w(x \to y) - dist(y) = \\
&= dist_0(x) + (0, -[\vec{h}], -\alpha) + w(x \to y) - dist_0(y) = \\
&= slack_0(x \to y) + (0, -[\vec{h}], -\alpha) \Rightarrow \\
&slack(x \to y) < 0, \text{ if } slack_0(x \to y) < (0, [\vec{h}], \alpha)
\end{aligned}$$

Furthermore, we need to take into account that $u \to v$ could potentially have non-trivial homology $[\vec{h}]$ and α term, in which case we need to define what it means to split uv to two separate edges us and sv, and how to resolve $[\vec{h}]$ and α terms. To deal with the issue, we reduce distances to u and v as follows:

$$w_\epsilon(s \to u) = (0, -[\vec{h}], -\alpha) \tag{9}$$
$$w_\epsilon(s \to v) = (1, [\vec{0}], 0) \tag{10}$$

This process can explained in following way:

1. When we split uv to 2 edges (us, sv), we assume $[\vec{h}], \alpha$ terms reside in sv side of uv at the start.
2. By reducing weight of $w(s \to v)$ by $(0, -[\vec{h}], -\alpha)$, we are removing homology and α terms from sv edge.
3. Then we add those values to su, which is equivalent to adding $(0, -[\vec{h}], -\alpha)$ to dart $w(s \to u)$.

Since we reduced distance to all vertices in the graph by equal amount (this is because each vertex is either child of u or v in T_s, so reducing distances to u and v is same as reducing distances to all vertices), the process does not introduce any pivots. We define a parametric weights as follows:

$$w_\lambda(s \to u) = (0, -[\vec{h}], -\alpha) - \lambda \tag{11}$$
$$w_\lambda(s \to v) = (1, [\vec{0}], 0) + \lambda \tag{12}$$

Every other dart $x \to y$ has constant parametric weight $w_\lambda(x \to y) = w(x \to y)$. 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 $\lambda = w(u \to v)$, $T_\lambda = T_v$.

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

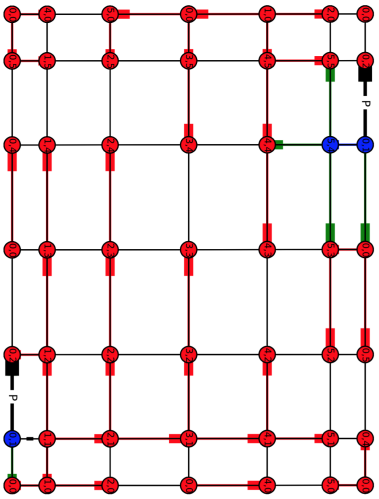
MoveAlongEdge$(G, u \to v, dist, pred)$:
  Add new vertex s
  $pred[u], pred[v] \leftarrow s$
  $\lambda \leftarrow 0$

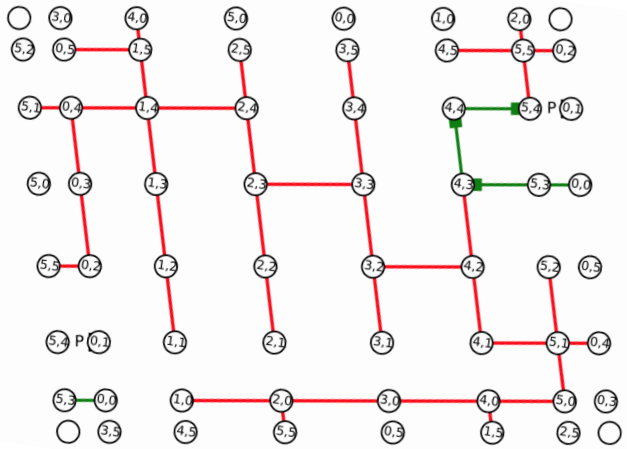
  $w(s \to u) \leftarrow (0, -[w(s \to u)], -\alpha(w(s \to u)))$
  AddSubtree$((0, -[w(s \to u)], -\alpha(w(s \to u))), u)$

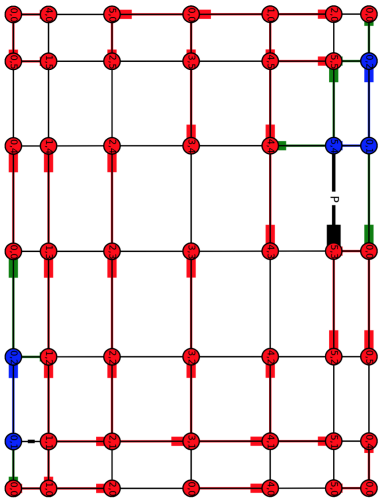
  $w(s \to v) \leftarrow (1, [\vec{0}], 0)$
  AddSubtree$((0, -[w(s \to u)], -\alpha(w(s \to u))), v)$

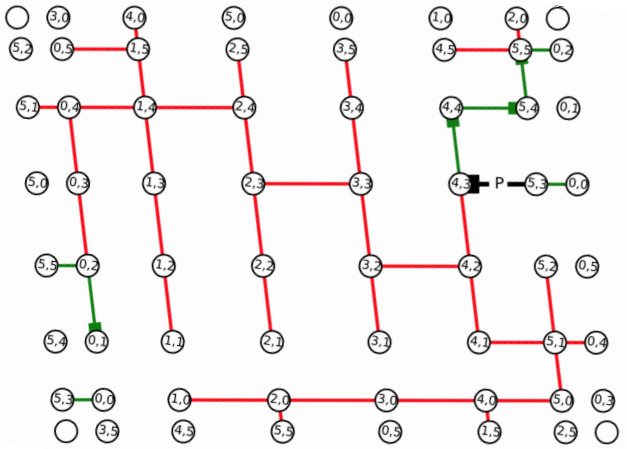
  while $\lambda < (1, [\vec{0}], 0)$:
    pivot $\leftarrow$ FindNextPivot
    If **pivot** is non NULL **AND** $(\lambda + slack(\textbf{pivot})/2) < (1, [\vec{0}], 0)$
      Pivot(**pivot**)
      $\lambda \leftarrow \lambda + slack(\textbf{pivot})/2$
    else
      $\delta = (1, [\vec{0}], 0) - \lambda$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

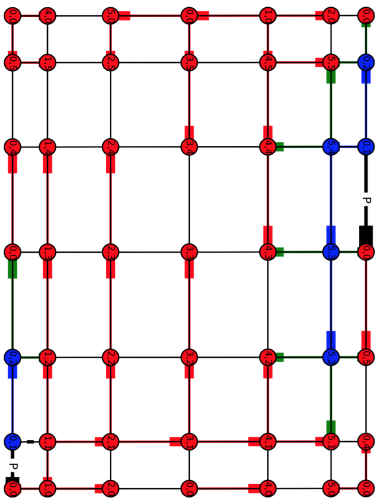
Below, we show an example of moving along an edge process on genus $g = 2$ grid.
On a left side, we see primal holy tree H with nodes increasing in distance are red, decreasing are blue, and next pivot is denoted as P.
On right side, we see the dual graph $(G/T)^*$ and active darts are noted with green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

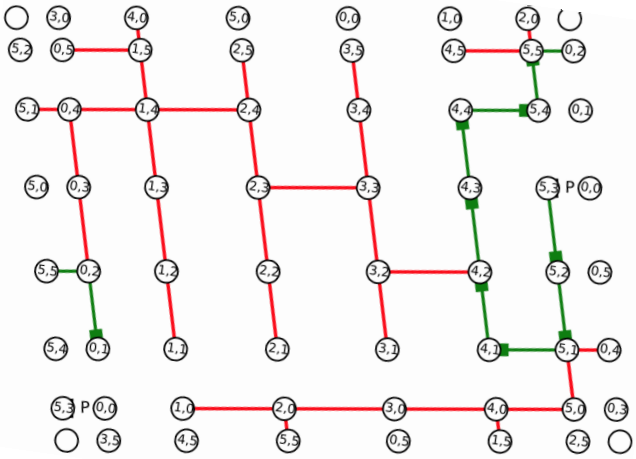












3 Bounding number of pivots

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: Suppose there is a crossing and let z be the last crossing point of P_i^y, P_j^y. $P_i^{z\to y}$, $P_j^{z\to y}$ be respective paths from z to y. According to the **Lemma 2.1**, there must be exactly one holiest path from z to y, which contradicts with definition of $P_i^{z\to y}$, $P_j^{z\to y}$ being distinct holiest paths from z to y. □

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

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

$$d \in T$$
$$d^* \in (G/T)^*$$
$$rev(d) \in T$$
$$rev(d^*) \in (G/T)^*$$

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 the holy tree and get replaced by a some edge in holy tree $O(g)$ times. □

4 Finding pivot quickly

Cabello et al.[5] use a grove data structure, a collection of separate trees with some vertices in original graph are replicated, to quickly find the next pivot. They maintain each tree in a dynamic tree and show that the total complexity of finding the next pivot is $O(g \log n)$.

- 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?

5 Analysis

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

References

[1] Glencora Borradaile and Philip Klein. "An O (n log n) algorithm for maximum st-flow in a directed planar graph". In: *Journal of the ACM (JACM)* 56.2 (2009), p. 9.

[2] Glencora Borradaile, Piotr Sankowski, and Christian Wulff-Nilsen. "Min st-cut oracle for planar graphs with near-linear preprocessing time". In: *ACM Transactions on Algorithms (TALG)* 11.3 (2015), p. 16.

[3] Glencora Borradaile et al. "Multiple-source multiple-sink maximum flow in directed planar graphs in near-linear time". In: *arXiv preprint arXiv:1105.2228* (2011).

[4] Sergio Cabello and Erin W Chambers. "Multiple source shortest paths in a genus g graph". In: *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics. 2007, pp. 89–97.

[5] 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.

[6] Erin W Chambers, Jeff Erickson, and Amir Nayyeri. "Homology flows, cohomology cuts". In: *SIAM Journal on Computing* 41.6 (2012), pp. 1605–1634.

[7] Erin W Chambers et al. "Splitting (complicated) surfaces is hard". In: *Proceedings of the twenty-second annual symposium on Computational geometry*. ACM. 2006, pp. 421–429.

[8] William H Cunningham. "A network simplex method". In: *Mathematical Programming* 11.1 (1976), pp. 105–116.

[9] 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.

[10] Jeff Erickson. "Maximum flows and parametric shortest paths in planar graphs". In: *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics. 2010, pp. 794–804.

[11] Jeff Erickson. "Combinatorial optimization of cycles and bases". In: *Proc. Sympos. Appl. Math.* Vol. 70. 2012, pp. 195–228.

[12] Jeff Erickson and Sariel Har-Peled. "Optimally cutting a surface into a disk". In: *Discrete & Computational Geometry* 31.1 (2004), pp. 37–59.

[13] Jeff Erickson and Kim Whittlesey. "Greedy optimal homotopy and homology generators". In: *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics. 2005, pp. 1038–1046.

[14] Monika R Henzinger et al. "Faster shortest-path algorithms for planar graphs". In: *journal of computer and system sciences* 55.1 (1997), pp. 3–23.

[15] Philip N Klein. "Multiple-source shortest paths in planar graphs". In: *SODA*. Vol. 5. 2005, pp. 146–155.

[16] Philip N Klein, Shay Mozes, and Oren Weimann. "Shortest paths in directed planar graphs with negative lengths: A linear-space O (n log 2 n)-time algorithm". In: *ACM Transactions on Algorithms (TALG)* 6.2 (2010), p. 30.

[17] Richard J Lipton and Robert Endre Tarjan. "A separator theorem for planar graphs". In: *SIAM Journal on Applied Mathematics* 36.2 (1979), pp. 177–189.

[18] Shay Mozes and Christian Wulff-Nilsen. "Shortest Paths in Planar Graphs with Real Lengths in O (nlog2 n/loglogn) Time". In: *Algorithms–ESA 2010*. Springer, 2010, pp. 206–217.

[19] Daniel D Sleator and Robert Endre Tarjan. "A data structure for dynamic trees". In: *Journal of computer and system sciences* 26.3 (1983), pp. 362–391.

[20] Robert E Tarjan. "Efficiency of the primal network simplex algorithm for the minimum-cost circulation problem". In: *Mathematics of Operations Research* 16.2 (1991), pp. 272–291.

[21] Robert E Tarjan. "Dynamic trees as search trees via euler tours, applied to the network simplex algorithm". In: *Mathematical Programming* 78.2 (1997), pp. 169–177.

[22] Robert E Tarjan and Renato F Werneck. "Self-adjusting top trees". In: *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics. 2005, pp. 813–822.

[23] Neal E Young, Robert E Tarjant, and James B Orlin. "Faster parametric shortest path and minimum-balance algorithms". In: *Networks* 21.2 (1991), pp. 205–221.