Using Hasse Diagrams to Compute a Gradient Vector Field

Brittany Fasy, Ben Holmgren, Brad McCoy, David Millman

January 28, 2020

Abstract

In computational geometry and topology, simplicial complexes are widely used structures that model abstract spaces. A simplicial complex is a grouping of n-dimensional generalizations of triangles, called simplices. Discrete Morse functions assign real numbers to components of simplicial complexes. These mappings are useful for modeling "heights" in an abstract space. One of the goals of Discrete Morse Theory is to generate gradient vector fields along the mappings of a function on a simplicial complex. These gradient vector fields use vectors on a simplicial complex to display how function values change throughout the complex. As a result, gradient vector fields can be thought of as indicating the "flow" of a simplicial complex. To add intuition, we can imagine a mountain range as a simplicial complex, with a corresponding Morse function acting as a map to the heights in that mountain range.

That complex's gradient vector field tells us where water would flow on that mountain range if we were to drop a bucket of water at any point. Constructing gradient vector fields is also extremely useful when studying the topology of a simplicial complex. However, we hypothesize that prevailing algorithms generate gradient vector fields with doubly exponential time complexity. We are proposing new algorithms which can compute gradient vector fields with improved time complexity. To do so, we have utilized Hasse Diagrams, a data structure which is convenient for storing a simplicial complex. Our algorithms make use of memoization techniques in a Hasse diagram in order to limit repetitive recursive calls prevalent in prior algorithms. Lastly, we are in the process of providing a thorough analysis of the time complexity of our algorithms alongside an analysis of current algorithms. In doing so, we expect to find novel contributions in understanding the time complexity of gradient vector field computation, alongside our addition of new algorithms to do so.

1 Introduction

A major goal of Discrete Morse Theory is to compute a gradient vector field on a simplicial complex given an injective function from the vertices of the complex to $\mathbb{R}$. Gradient vector fields enable one to compute the homology groups of a simplicial complex efficiently. In this work we analyze the runtime of an algorithm given by [**?**] and restated in [**?**]. Our main result is that we show the runtime can be improved by utilizing basic dynamic programming techniques.

To achieve this result we work almost entirely with the Hasse Diagram of the simplicial complex. Specifically, if we consider the Hasse Diagram for a simplicial complex, we can order that Hasse Diagram based on the function values for each simplex. Using these ordered Hasse Diagrams, one is able to see the unnecessary steps in the algorithm of [**?**]. To begin, we will provide a formal analysis of the time complexity of current gradient vector field algorithms, and outline their computation by way of Hasse Diagrams. Then, we will outline our primary result, an improved gradient vector field algorithm using a Hasse Diagram, and provide a formal time complexity analysis for this new contribution.

2 Background

In this section we provide definitions and notation that will be used throughout the paper. In general our notation follows that of [**?**]. Let M be a simplicial complex. We denote a typical $p-$simplex by σ^p or σ if the dimension is clear.

Definition 2.1. A function $f : M \to \mathbb{R}$ is a *discrete Morse function*, if for every $\sigma^p \in M$, the following two conditions hold:

1. $|\{\beta^{(p+1)} > \sigma | f(\beta) \leq f(\sigma)\}| \leq 1$,
2. $|\{\gamma^{(p-1)} < \sigma | f(\gamma) \geq f(\sigma)\}| \leq 1$.

A nice intuitive definition is given in [**?**], "the function generally increases as you increase the dimension of the simplices. But we allow at most one exception per simplex."

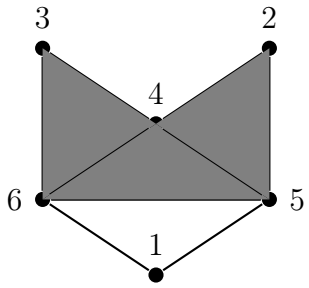
Definition 2.2. Let $f : M \to \mathbb{R}$ be a discrete Morse function. A simplex σ^p is *critical* if the following two conditions hold:

1. $|\{\beta^{(p+1)} > \sigma | f(\beta) \leq f(\sigma)\}| = 0$,
2. $|\{\gamma^{(p-1)} < \sigma | f(\gamma) \geq f(\sigma)\}| = 0$.

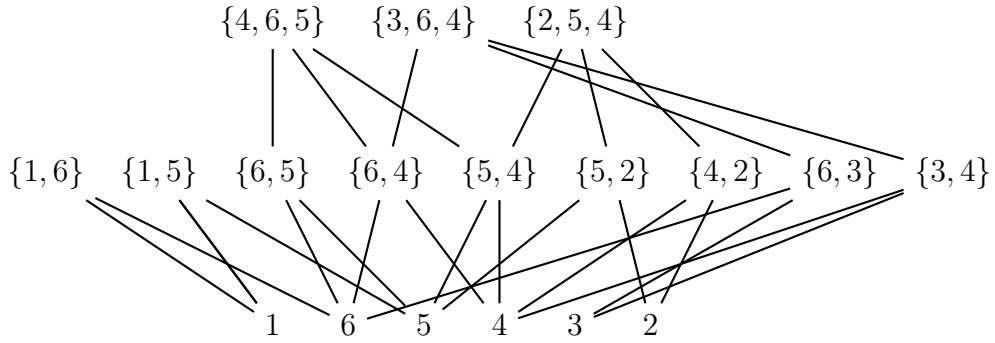
Definition 2.3. Let $\mathbb{H}_K$ be the Hasse diagram of a simplicial complex K, where $\mathbb{H}_M$ is a graph with a 1-1 correspondence between the nodes of $\mathbb{H}$ and the simplices of K. We denote a simplex as $\sigma \in K$ or $\sigma \in \mathbb{H}$ for simplices of K and the nodes of $mathbbH$ respectively.

Definition 2.4. We define an ordered Hasse diagram $\mathbb{H}^*_K$ of a simplicial complex K as the sorted $\mathbb{H}_K$ where function values increase from left to right. The ordered Hasse diagram is l exicographically sorted, so the function values of its "nondominant "components are also sorted and increase from left to right.

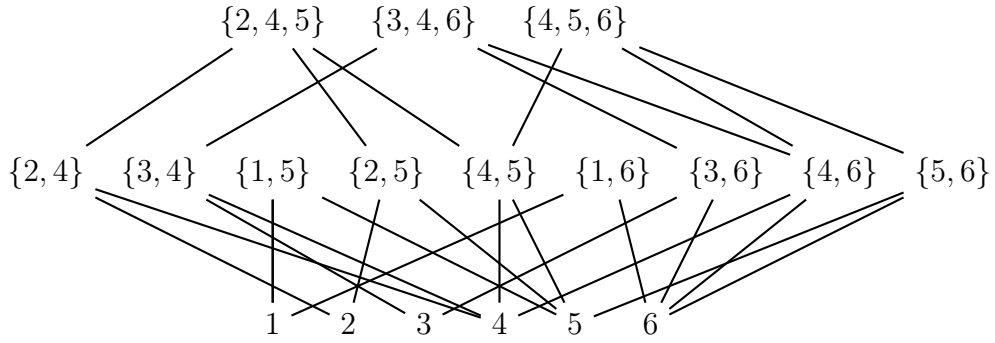
Example 2.4



Equivalent Hasse Diagram $\mathbb{H}_K$



Equivalent Ordered Hasse Diagram $\mathbb{H}_K^*$



3 Runtime

In this section we analyze the runtime of the algorithm given in [**?**]. Notice that any simplicial complex$K$ with d vertices is contained in the complete simplex consisting of d vertices which we denote Δ^d. Thus the runtime of an algorithm on K is less than or equal to the runtime on Δ^d. Let n be the number of simplicies in K. There are 2^d simplicies in Δ^d.

The algorithm Extract computes the lower link of each vertex in the simplex.

Lemma 3.1 (lower link)**.** *The lower link of each vertex can be computed in* $O(n)$.

Proof. For a given vertex, v assign all simplicies that contain v the color blue. Then assign all subsets of the blue simplicies the color red. The lower link of v is all simplicies that are red and not blue and have value less than the value of v. □

Corollary 3.2. *The lower links of all vertices can be computed in* $O(n \cdot n_0)$.

We now consider an algorithm that computes the lower link of each vertex in K, we call this algorithm LowerLink(K). LowerLink(K) is contained in extract raw, specifically in lines 1 through 8 in [**?**].

Algorithm 1 LowerLink(K)

```
1: Input: K a simplicial complexwith n_0 vertices.
2: Output: The lower link of each vertex.
3: for all v ∈ K Do
4: Let K' := the lowerlink of v
5: if K' ≠ then
6:     Return Lowerlink(K').
7: end if
```

Lemma 3.3 (no memoization)**.** *The runtime of LowerLink*(Δ^d) *without memoization is* $O(d^2 \cdot n)$.

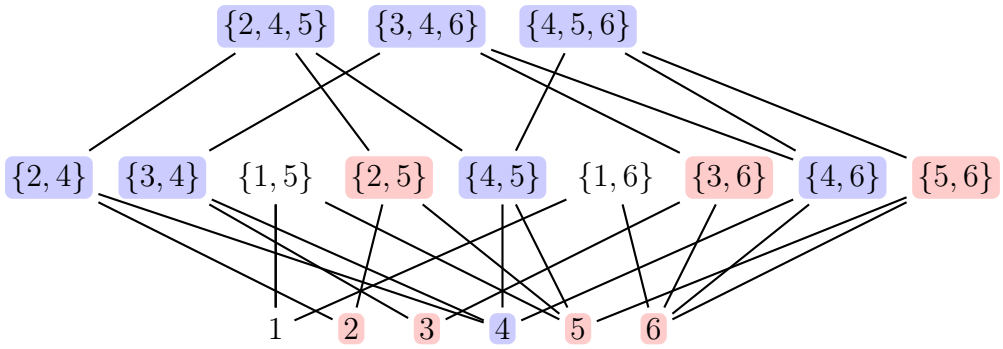
Proof. Let K be a simplicial complex containing n_0 vertices, and let Δ^{n_0} be a complete simplicial complex which contains n_0 vertices. $K \subseteq \Delta^{n_0}$, so in the "worst" case, we are concerned with computing the lower link of a complete simplicial complex with n_0 vertices. For the Hasse Diagram $\mathbb{H}_K$ equivalent to K, let $n = \sum_{n=0}^{dim(K)} n_i$ where n is the total number of nodes in $\mathbb{H}_K$. To compute the lower link, we must "march" up the Hasse diagram from a simplex, and then pass by all of the children of the selected highest degree simplices. This will occur in $O(n + n) = O(n)$ time. □

Lemma 3.4 (yes memoization)**.** *The runtime of LowerLink*(Δ^d) *with memoization is* $O(d{\cdot}n)$.

Proof. □

Example 3.1 Ordered Hasse Diagram $\mathbb{H}_K^*$ For K in Example 2.4

Compute Lower Link of $f^{-1}(4)$:



Thus, the lower link of $f^{-1}(4) = \{f^{-1}(2), f^{-1}(3)\}$.

Algorithm 2 Extract Raw Analogous

Input: Simplicial Complex K, injective $f_0 : V(K) \to \mathbb{R}$
Output: Gradient vector field $(A, B, C, r : B \to A)$ on K
$K_H \leftarrow$ Hasse Diagram for K
Sort K_H lexicographically
for $i = Dim(K)$; $i \geq 0$; $i--$ **do** :
    for $\alpha \in K[i]$ **do** :
        if α is unmarked and rightmost child is unmarked **then**
            Add rightmost (largest lexicographical value) child to B
            Add α to A
        end if
    end for
end for
if $\exists\ \alpha \in K$ which are unmarked **then**
    Add all remaining unmarked α to C
end if

Algorithm 3 Extract Cancel Analogous

Input: Ordered Hasse Diagram K_H, Gradient vector field $(A, B, C, r : B \to A)$ on K
Output: "Refined" Gradient vector field $(A, B, C, r : B \to A)$ on K
for $\alpha \in C$ **do** :
    for all σ which are parent nodes to $\alpha \in K_H$ **do**
        mark all child nodes of σ
    end for
    if any marked nodes intersect with a node marked from a previous α **then**
        Compare the gradient of current α and its max child node to the same gradient of intersecting α
        Consider the α which composes a larger gradient (larger difference between min and max weights of child nodes), add max child node of this α to B and add this α to A
    end if
end for
if any α remain whose siblings have no intersections with the siblings of any other α **then**
    Add each of these α to A and its corresponding max child node to B
end if
Return: "Revised" Gradient Vector Field (A,B,C,$r : B \to A$ on K