If You Must Choose Among Your Children, Pick the Right One

Brittany Terese Fasy[†*] Benjamin Holmgren[†] Bradley McCoy[†] David L. Millman[†]

Abstract

Given a simplicial complex K and an injective function f from the vertices of K to $\mathbb{R}$, we consider algorithms that extend f to a discrete Morse function on K. We show that an algorithm of King, Knudson and Mramor can be described on the directed Hasse diagram of K. Our description has a faster runtime for high dimensional data with no increase in space.

1 Introduction

Milnor's classical Morse theory provides tools for investigating the topology of smooth manifolds [16]. In [9], Forman showed that many of the tools for continuous functions can be applied in the discrete setting. Inferences about the topology of a CW complex can be made from the number of critical cells in a Morse function on the complex.

Given a Morse function one can interpret the function in many ways. Switching interpretations is often revealing. In this paper, we think of a discrete Morse function in three different ways. Algebraically, a Morse function is a function from the faces of a complex to the real numbers, subject to certain inequalities. Topologically, a Morse function is a pairing of the faces such that the removal of any pair does not change the topology of the complex. Combinatorially, a Morse function is an acyclic matching in the Hasse diagram of the complex, where unmatched faces correspond to critical cells.

Discrete Morse theory can be combined with persistent homology to analyze data, see [1, 2, 4, 5, 7, 8, 13]. When dealing with data, we have the additional constraint that vertices have function values assigned. For complexes without any preassigned function values, Joswig and Pfetsch showed that finding a Morse function with a minimum number of critical cells is NP-Hard [12]. Algorithms that find Morse functions with relatively few critical cells have been explored in [11,15,17].

In this work, we consider the algorithm Extract, Algorithm 1, given in [13]. Extract takes as input a simplicial complex and an injective function from the vertices to the reals, and returns a discrete Morse function, giving topological information about the complex. We show that a subalgorithm of Extract, ExtractRaw can be simplified by considering the directed Hasse diagram. This simplification leads to an improved runtime and no change in space. The paper is organized as follows, in Section 2, we provide the definitions that will be used in the paper. In Section 3, we describe Extract and analyze the runtime, then, in Section 4, we give our reformulation and show that the runtime is improved from $\Omega(n^2 \log n)$ to $O(dn)$ where n is the number of cells and d is the dimension of K.

*Department of Mathematical Sciences, Montana State U.
†School of Computing, Montana State U.
{brittany.fasy, bradleymccoy, david.millman}@montana.edu
benjamin.holmgren@student.montana.edu

2 Background

In this section, we provide definitions, notation, and primitive operations used throughout the paper. For a general overview of discrete Morse theory see [14, 19], note that both texts provide a description of Extract originally given in [13]. Extract is the starting point for this work.

In what follows, we adapt the notation of Edelsbrunner and Harer [6] to the definitions of Forman [10]. Here, we work with simplicial complexes, but the results hold for CW complexes. Let K be a simplicial complex with n simplices. For $i \in \mathbb{N}$, denote the i-simplices of K as K_i, the number of simplices in K_i as n_i, and the dimension of the highest dimensional simplex of K as $\dim(K)$.

Let $\sigma \in K$, denote the dimension of σ as $\dim(\sigma)$. Let $p = \dim(\sigma)$ and $\{v_0, v_1, \cdots, v_p\} \subseteq K_0$ be the zero-simplices of σ, then we say $\sigma = [v_0, v_1, \cdots, v_p]$. If $\tau \in K$ is disjoint from σ, then we can define the *join* of σ and τ to be the $(\dim(\sigma) + \dim(\tau) + 1)$-simplex that consists of the union of the vertices in σ and τ, denoted $\sigma * \tau$. We write $\sigma \prec \tau$ if σ is a proper face of τ.

Let $p \in \mathbb{N}$ and consider simplices $\sigma_u, \sigma_v \in K_p$, with $\sigma_u = [u_0, u_1, \ldots, u_p]$ and $\sigma_v = [v_0, v_1, \ldots, v_p]$. Let $f_0 : K_0 \to \mathbb{R}$ be an injective function. Without loss of generality, assume that the zero-simplices of σ_u and σ_v are sorted by function value, that is, we have $f_0(v_i) < f_0(v_j)$ when $0 \le i < j \le p$, similarly for σ_u. We say that σ_u is lexicographically smaller than σ_v, denoted $\sigma_u <_{lex} \sigma_v$, if the vector $\langle f_0(u_p), f_0(u_{p-1}), \ldots, f_0(u_0)\rangle$ is lexicographically smaller than $\langle f_0(v_p), f_0(v_{p-1}), \ldots, f_0(v_0)\rangle$.

The *star* of v in K, denoted $\mathrm{star}_K(v)$, is the set of all simplices of K containing v. The *closed star* of v in K, denoted $\overline{\mathrm{star}}_K(v)$, is the closure of $\mathrm{star}_K(v)$. The *link* of v in K, is denoted as $\mathrm{link}_K(v) := \overline{\mathrm{star}}_K(v) \setminus \mathrm{star}_K(v)$. We define the *lower link* of v, denoted $\mathrm{lowerlink}_K(v)$,

to be the maximal subcomplex of $\text{link}_K(v)$ whose zero-simplices have function value less than $f_0(v)$; the lower link can be computed in $O(n)$ time.

We provide the definition of a Morse function, modified from Forman [10].

Definition 1 (Morse Function) *A function $f : K \to \mathbb{R}$ is a* discrete Morse function*, if for every $\sigma \in K$, the following two conditions hold:*

1. $|\{\beta \succ \sigma | f(\beta) \leq f(\sigma)\}| \leq 1$,
2. $|\{\gamma \prec \sigma | f(\gamma) \geq f(\sigma)\}| \leq 1$.

An intuitive definition is given in [19], "the function generally increases as you increase the dimension of the simplices. But we allow at most one exception per simplex." Let $f : K \to \mathbb{R}$ be a discrete Morse function. A simplex $\sigma \in K$ is *critical* if the following two conditions hold:

1. $|\{\beta \succ \sigma | f(\beta) \leq f(\sigma)\}| = 0$,
2. $|\{\gamma \prec \sigma | f(\gamma) \geq f(\sigma)\}| = 0$.

Simplices that are not critical are called *regular*.

Given a discrete Morse function f on a simplicial complex K, we define the induced *gradient vector field*, or GVF, for short, as $\{(\sigma, \tau) : \sigma \prec \tau, f(\sigma) \geq f(\tau)\}$. Note that σ is a codimension one face of τ. We can gain some intuition for this definition by drawing arrows on the simplicial complex as follows. If σ is regular, a codimension one face of τ, and $f(\tau) \leq f(\sigma)$, then we draw an arrow from σ to τ. Constructing a GVF for a simplicial complex is as powerful as having a discrete Morse function, and is the goal of both Extract and our proposed Algorithm 2.

Next, we define two functions that are helpful when constructing a GVF. The *rightmost face* of σ, denoted $\rho(\sigma)$, is the face of σ with maximum lexicographic value. The *leftmost coface* of σ, denoted $\ell(\sigma)$, is the dimension one coface of σ with minimum lexicographic value. We say σ is a *left-right parent* and we call $\rho(\sigma)$ a *left-right child* if $\ell \circ \rho(\sigma) = \sigma$.

In [10], Forman showed that each simplex in K is exclusively a tail, head, or unmatched. Moreover, the unmatched simplices are critical. Thus, we can partition the simplices of K into *heads* H, *tails* T, and *critical simplices* C, and encode the GVF as a bijection $m : T \to H$. That is, we can represent the GVF for f as the unique tuple (H, T, C, m). We will use this representation throughout our algorithms.

Note that a GVF is a particularly useful construction. It provides a way to reduce the size of a simplicial complex without changing the topology (by cancelling matched pairs), which is constructive for preprocessing large simplicial complexes. See [4, 17] for examples.

We define a consistent GVF as follows:

Definition 2 (Consistent GVF) *Let K be a simplicial complex, and let $f_0 \colon K_0 \to \mathbb{R}$ be injective. Then, we say that a gradient vector field (H, T, C, m) is* consistent *with f_0 if, for all $\varepsilon > 0$, there exists a discrete Morse function $f \colon K \to \mathbb{R}$ such that*

(a) (H, T, C, m) is the GVF corresponding to f.

(b) $f|_{K_0} = f_0$.

(c) $|f(\sigma) - \max_{v \in \sigma} f_0(v)| \leq \varepsilon$.

Let (H, T, C, m) be a GVF. Then, for $r, p \in \mathbb{N}$, a *gradient path*[1] is a sequence of simplices in K:

$$\Gamma = \{\sigma_{-1}, \tau_0, \sigma_0, \tau_1, \sigma_1, \ldots, \tau_r, \sigma_r, \tau_{r+1}\}$$

beginning and ending with critical simplices $\sigma_{-1} \in K_{p+1}$ and $\tau_{r+1} \in K_p$ such that for $0 \leq i < r$, $\tau_i \in K_p$, $\sigma_i \in K_{p+1}$, $m(\tau_i) = \sigma_i$, and $\tau_i \succ \sigma_{i+1} \neq \sigma_i$. We call a path *nontrivial* if $r > 0$.

3 A Discrete Morse Extension of $f_0 \colon K_0 \to \mathbb{R}$

In this section, we give a description of Algorithm 1 (Extract), originally from [13]. This algorithm takes a simplicial complex K, an injective function $f_0 \colon K_0 \to \mathbb{R}$, and a threshold that ignores pairings with small persistence $p \geq 0$; and returns a GVF on K that is consistent with f_0.

Algorithm 1 [13] Extract

Input: A simplicial complex K, injective function $f_0 \colon K_0 \to \mathbb{R}$, and $p \geq 0$
Output: a GVF consistent with f_0

```
1: γ ← ExtractRaw(K, f_0)                          ▷ Algorithm 3
2: for j = 1, 2, ..., dim(K) do
3: |   γ ← ExtractCancel(K, h, p, j, γ)             ▷ Alg. 4
4: return γ
```

Extract uses two subroutines: First, in Line 1 of Algorithm 1 ExtractRaw (given in Algorithm 3) is used to generate an initial GVF on K consistent with f_0. Let (H_0, T_0, C_0, r_0) be this initial GVF. Then, for each dimension ($j = 1$ through $\dim(K)$), the algorithm makes a call to ExtractCancel (given in Algorithm 4) that augments an existing gradient path to remove simplices from C_0 in pairs. For more details, see Appendix A.1.

In the next section, we provide a simpler and faster algorithm to replace ExtractRaw, which dominates the runtime of Extract when $p = 0$ (and in practice, when p is very small). We conclude this section with properties of the output from ExtractRaw:

[1] There is a slight discrepancy between the definition of Forman [10] and KKM [13]. In particular, Forman's definition states the head and the tail of the path are simplices of the same dimension. On the other hand, KKM's usage in the ExtractCancel algorithm expects that the head and tail are different dimensions. Here, we state the definition implied by the usage in KKM.

Theorem 3 (Properies of ExtractRaw) *Let K be a simplicial complex, let $f_0 \colon K_0 \to \mathbb{R}$ be an injective function, and suppose (H, T, C, m) is the output of* ExtractRaw(K, f_0). *Let $\varepsilon > 0$. Then, there exists a discrete Morse function $f \colon K \to \mathbb{R}$ such that the following hold:*

(i) (H, T, C, m) is a GVF consistent with f_0.

(ii) Let $\sigma \in K$. Then, $\sigma \in H$ if and only if σ is a left-right parent.

(iii) For all $\sigma \in H$, $m(\rho(\sigma)) = \sigma$.

(iv) The runtime of ExtractRaw *is $\Omega(n^2 \log n)$.*

4 A Faster Algorithm for ExtractRaw

The main contribution of this paper is ExtractRightChild, which we show is a simplified version of ExtractRaw that has the same output with an improved runtime. This section provides a description of the algorithm, and a proof of the equivalence with ExtractRaw.

4.1 Hasse Diagram Data Structure

We assume that KKM [13] represent K in a *standard Hasse diagram data structure* $\mathcal{H}$, which can be encoded as an adjacency list representation for a graph. Each simplex $\sigma \in K$ is represented by a node in $\mathcal{H}$. We abuse notation and write $\sigma \in \mathcal{H}$ as the corresponding node. Two simplices $\sigma, \tau \in \mathcal{H}$ are connected by an edge from σ to τ if σ is a codimension one face of τ. For a node $\sigma \in \mathcal{H}$, we partition its edges into two sets, $up(\sigma)$ and $down(\sigma)$ as the edges in which σ is a face or coface, respectively.

For $p \in \mathbb{N}$, we denote the nodes of $\mathcal{H}$ corresponding to the p-simplices of K as $\mathcal{H}_p$ and we store each $\mathcal{H}_p$ in its own set that can be accessed in $O(1)$ time. Note that there is no requirement about the ordering of the edges or the nodes in each $\mathcal{H}_p$. See Figure 1 for an example of the data structure.

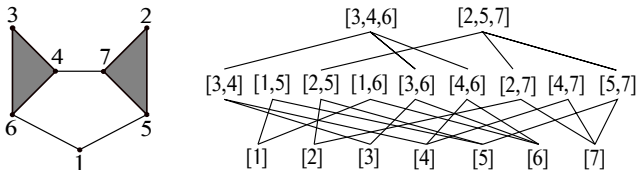


Figure 1: *Left*: A simplicial complex with function values assigned to the vertices. *Right*: The Hasse diagram of the simplicial complex.

For our algorithm, we decorate each node of $\mathcal{H}$ with additional data. For clarity, we denote the decorated data data structure as $\mathcal{H}^*$. Next, we describe the additional data stored in each node and how to initialize the data. Consider $\sigma \in \mathcal{H}_p^*$ and define $\tilde{f}(\sigma) := \max_{v \in \sigma} f_0(v)$. Each node stores $\tilde{f}(\sigma)$, the rightmost child $\rho(\sigma)$ and leftmost parent $\ell(\sigma)$.

Next, we describe how to initialize the data and summarize with the following lemma.

Lemma 4 (Hasse decoration) *Given a simplicial complex K with n simplices and $\dim(K) = d$. The decorated Hasse diagram uses $O(n)$ additional space. We can decorate the Hasse digram K in $O(dn)$ time.*

Proof. We begin by analyzing the space complexity. For each node, we store a constant amount of additional data. Thus, the decorated Hasse diagram uses $O(n)$ additional space.

Next, we analyze the time complexity. To decorate $\mathcal{H}$ for each node $\sigma \in \mathcal{H}$, we must compute $\tilde{f}(\sigma)$, $\rho(\sigma)$, and $\ell(\sigma)$. Let $p = \dim(\sigma)$. We proceed in three steps.

First we compute $\tilde{f}$. In general, computing $\tilde{f}(\sigma)$ takes $O(p)$ time, since there may be no more than p vertices which compose any σ. Let τ_1 and τ_2 be distinct codimension one faces of σ. Observe that $\tilde{f}(\sigma) = \max(\tilde{f}(\tau_1), \tilde{f}(\tau_2))$. Thus, if we know the function values for $\mathcal{H}_{p-1}$, we can compute and store all function values of all nodes in $\mathcal{H}$ in $\Theta(n)$ time.

Second we compute $\rho(\sigma)$ by brute force. We iterate over all edges in $down(\sigma)$ to find its largest face under lexicographic ordering. Since a p-simplex σ has $p + 1$ down edges, computing $\rho(\sigma)$ for σ takes $O(p)$ time. As $\dim(K) = d$, and there are n nodes, we can then compute ρ for all nodes in $O(dn)$ time.

Third, we compute ℓ, also by brute force. We iterate over all edges in $up(\sigma)$ to find its smallest lexicographical coface. While we cannot bound $up(\sigma)$ as easily as $down(\sigma)$, we do know that when computing ℓ we can charge each edge in the Hasse diagram for one comparison. Observe that when computing ρ, we can similarly charge each comparison to an edge. Then, from computing ρ, we know the total number of comparisons is $O(dn)$. Thus, the total number of comparisons for computing ℓ is also $O(dn)$.

As each step takes $O(dn)$ time, decorating $\mathcal{H}$ takes $O(dn)$ time. □

4.2 Algorithm Description

Next, we describe the main algorithm. Given a simplicial complex K (represented as a Hasse diagram), and an injective function $f_0 : K_0 \to \mathbb{R}$, ExtractRightChild computes a GVF consistent with f_0.

Algorithm 2 has three main steps. First, we create a decorated Hasse diagram. Second, we process each level of the Hasse diagram from top to bottom. For

Algorithm 2 ExtractRightChild

Input: simp. complx. K, injective fcn. $f_0 : K_0 \to \mathbb{R}$
Output: a GVF consistent with f_0

1: $\mathcal{H}^* \leftarrow$ decorate the Hasse diagram of K ▷ Lem. 4
2: $T \leftarrow \emptyset$, $H \leftarrow \emptyset$, $C \leftarrow \emptyset$, $m \leftarrow \emptyset$
3: **for** $i = \dim(K)$ to 1 **do**
4: **for** $\sigma \in \mathcal{H}_i^*$ **do**
5: **if** σ is assigned **then**
6: **continue**
7: **if** σ is a left-right parent **then**
8: Add $\rho(\sigma)$ to T; Add σ to H
9: Add $(\rho(\sigma), \sigma)$ to m
10: Mark σ and $\rho(\sigma)$ as assigned
11: **else**
12: Add σ to C
13: Mark σ as assigned
14: Add any unassigned zero-simplices to C
15: **return** (T, H, C, r)

each unassigned simplex, we check for a left-right parent node, and use the results to build up a GVF. Third, we process unassigned zero-simplices. See Figure 2 for an example.

4.3 Analysis of ExtractRightChild

For the remainder of this section, we prove that Algorithm 2 (ExtractRightChild) is equivalent to and faster than Algorithm 3 (ExtractRaw). For the following lemmas, let K be a simplicial complex, let $f_0 \colon K_0 \to \mathbb{R}$ be an injective function, and let (H, T, C, m) be the output of ExtractRightChild(K, f_0).

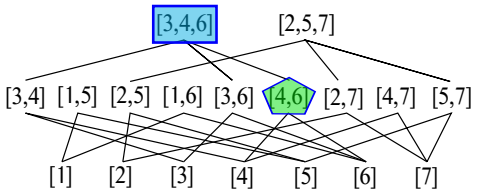
First, we show that (H, T, C, m) is a partition of K.

Lemma 5 (Partition) *The sets H, T, and C partition K.*

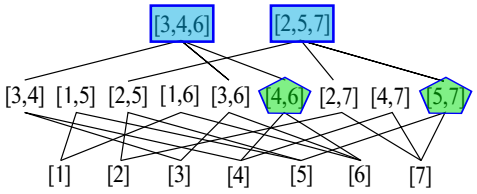
Proof. By Line 3 and Line 4 of Algorithm 2, ExtractRightChild iterates over all $\sigma^d \in K$ with $d > 0$ once. Each σ is either assigned or unassigned. If σ is unassigned, there are two options; σ may be a left-right parent, or it may not be. If σ is a left-right parent, Line 8 ensures that σ is put into H. Otherwise, Line 12 ensures that σ is put into C. If σ is assigned, then σ was assigned to T in Line 8. Thus, every $\sigma^d \in K$ with $d > 0$ must be assigned to exactly one of H, T, or C. Then, every σ^0 is again either assigned or unassigned. If assigned, $\sigma^0 \in T$. If unassigned, σ^0 is added to C in Line 14. Thus, every $\sigma \in K$ is assigned one of H, T, or C, making H, T, and C partition K. □

We will show that (H, T, C, m) satisfies (i), (ii), and (iii) of Theorem 3. Later in this section, we show that any GVF with these properties is unique.

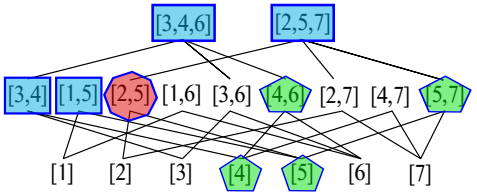
First, we show (iii) and one direction of (ii).



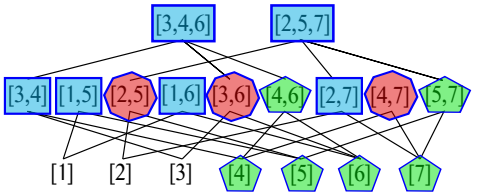
(a) The simplex $[3, 4, 6]$ is a left-right parent because $\ell \circ \rho([3, 4, 6]) = \ell(4, 6) = [3, 4, 6]$. The algorithm adds $[3, 4, 6]$ to H and $[4, 6]$ to T.



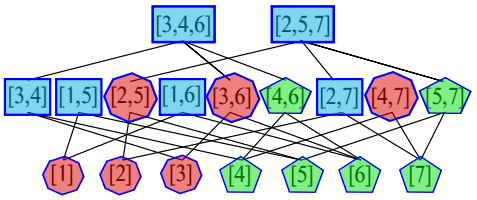
(b) The simplex $[2, 5, 7]$ is also a left-right parent. The algorithm adds $[2, 5, 7]$ to H and $[5, 7]$ to T.



(c) Both $[3, 4]$ and $[1, 5]$ are left-right parents. The simplex $[2, 5]$ is not a left-right parent because $\ell \circ \rho([2, 5]) = [1, 5] \neq [2, 5]$. The algorithm adds $[2, 5]$ to C.



(d) The loop in Line 3 is complete.



(e) The algorithm adds unassigned vertices to C.

Figure 2: Here we see a visualization of Algorithm 2, ExtractRightChild on the complex shown in Figure 1. Algorithm 2 partitions the nodes of the Hasse diagram into three sets, H, T and C. Elements of H are represented by blue rectangles, elements of T by green pentagons and elements of C by red hexagons.

Lemma 6 (Child Heads are Parents) *Let $\sigma \in H$. Then, σ is a left-right parent and $m(\rho(\sigma)) = \sigma$.*

Proof. Recall that H is the second output of ExtractRightChild(K,f_0), given in Algorithm 2. As Line 8 is the only step in which simplices are added to H and is within an if statement that checks if σ is a left-right parent, σ must be a left-right parent. Also within the if statement, Line 9 adds $(\rho(\sigma), \sigma)$ to m, which means that $m(\rho(\sigma)) = \sigma$. □

Now we show the reverse direction of (ii).

Lemma 7 (Child Parents are Heads) *Let $\sigma \in K$. If σ is a left-right parent, then $\sigma \in H$.*

Proof. Recall that in order for σ to be a left-right parent, we must have $\ell(\rho(\sigma)) = \sigma$. Now, we consider two cases. For the first case, suppose $\sigma \in C$. Then σ is added to C in Line 12 of Algorithm 2 when $\rho(\sigma)$ must already be assigned to $h <_{lex} \sigma$. So, $\ell(\rho(\sigma)) = h \neq \sigma$ and σ is not a left-right parent.

For the second case, suppose $\sigma = [v_0, v_1, \ldots, v_d] \in T$. Then σ is added to T in Line 8 of Algorithm 2 where $\sigma = \rho(h)$ for some $h = [v_{-1}, v_0, \ldots, v_d] \in H$ with $f_0(v_{-1}) < f_0(v_0)$. Notice that $\xi = [v_{-1}, v_1, v_2, \ldots, v_d]$ is a face of h and $\xi <_{lex} \sigma$. Then, $\ell(\rho(\sigma)) = \ell([v_1, \ldots, v_d]) \leq_{lex} [v_{-1}, v_1, v_2, \ldots, v_d] <_{lex} [v_0, v_1, \ldots, v_d] = \sigma$ and σ is not a left-right parent.

Thus, if σ is a left-right parent, then $\sigma \in H$.

□

To see (H, T, C, m) satisfies (i) we have the following lemma:

Lemma 8 (Consistency) *The tuple (H, T, C, m) is a gradient vector field consistent with f_0.*

Proof. Let $\varepsilon > 0$ and $d = \dim(K)$. Let $(H, T, C, m) =$ ExtractRightChild(K,f_0). We define

$$\delta := \min\{\varepsilon, \min_{v,w \in K_0} |f(v) - f(w)|\}.$$

We define $f\colon K \to \mathbb{R}$ recursively as follows: for all vertices $v \in K_0$, define $f(v) := f_0(v)$. Now, assume that f is defined on the i-simplices, for some $i \geq 0$. For each $\sigma \in K_{i+1}$, we initially assign $f(\sigma) = \max_{\tau \prec \sigma} f(\tau)$, then we update:

$$f(\sigma) = f(\sigma) + \begin{cases} -2^{1-2i}\delta & \text{if } \sigma \text{ is a left-right parent;} \\ 2^{1-2i}\delta & \text{otherwise,} \end{cases} \tag{1}$$

where j is the index of σ in the lexicographic ordering of all simplices. We make one final update:

$$f(\sigma) = f(\sigma) + \begin{cases} 2^{-2i}\delta & \text{if } \sigma \text{ is a left-right child;} \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

We need to show that (H, T, C, m) and f satisfy the three properties in Definition 2.

First, we show Part (a) of Definition 2 holds for f as defined above (that (H, T, C, m) is the GVF corresponding to f). Let (H', T', C', m') be the GVF corresponding to f. Since H, T, C partitions K by Lemma 5, it suffices to show that m is a bijection and $m = m'$. The only time that simplices are added to H or T happens directly alongside when pairs are added to m in lines 8 and 9, forcing that m must be a match.

Let $(\tau, \sigma) \in m$. Let $i = \dim(\sigma)$. By Lemma 7, σ is a left-right parent and $\tau = \rho(\sigma)$, which means that (τ, σ) is a left-right pair. We follow the computation of $f(\sigma)$. Since (τ, σ) is a left-right pair, τ is the rightmost face of σ, which means $f(\sigma)$ is initialized to $f(\tau)$. Since σ is a left-right parent, $f(\sigma)$ is updated by (1) to $f(\sigma) = f(\sigma) - 2^{1-2i}\delta$. Since σ is not a left-right child, nothing changes in (2). Thus, $f(\sigma) < f(\tau)$. Next, let $\tau' \prec \sigma$ such that $\tau' \neq \tau$ and $\dim(\tau') = i - 1$. We follow the computation of $f(\tau')$. Since τ is the only face of σ that is a left-right child, for any other $\tau' \prec \sigma$, (2), adds zero to the definition of $f(\tau')$. Recalling that (2) adds $2^{-2(i-1)}\delta$ to the definition of $f(\tau)$, we find that $f(\tau) \geq f(\tau') + 2^{-2(i-1)}\delta$, and

$$f(\sigma) = f(\tau) - 2^{1-2i}\delta \geq f(\tau') + 2^{-2(i-1)}\delta - 2^{1-2i}\delta \geq f(\tau').$$

Because σ may be any arbitrary left-right parent, we can guarantee that the above inequality is valid for any $(\sigma, \tau) \in m$ when related to any other faces of σ. Thus, f is discrete Morse, since it is impossible for f to violate the inequality given in Definition 1.

Since $f(\tau) > f(\sigma)$ and f is a discrete Morse function, we obtain $(\tau, \sigma) \in m'$. Each of these statements are biconditional, so we have shown that $m = m'$.

Part (b) of Definition 2 ($f|_{K_0} = f_0$) holds trivially.

Finally, we show Part (c) of Definition 2 holds (that $|f(\sigma) - \max_{v \in \sigma} f_0(v)| \leq \varepsilon$). By construction,

$$|f(\sigma - \max_{v \in \sigma} f_0(v)| \leq \left(\sum_{i=1}^{d} 2^{-i}\right)\delta = (1 - 2^{-d})\delta < \varepsilon.$$

□

Properties (i), (ii), and (iii) are quite restrictive. In fact, they uniquely determine a GVF, as we now show.

Theorem 9 (Unique GVF) *Let K be a simplicial complex and let $f_0\colon K_0 \to \mathbb{R}$ be an injective function. There is exactly one gradient vector field, (H, T, C, m), with the following two properties:*

(i) (H, T, C, m) is consistent with f_0.

(ii) For all $\sigma \in K$, $\sigma \in H$ if and only if σ is a left-right parent.

(iii) For all $\sigma \in H$, $m(\rho(\sigma)) = \sigma$.

Proof. Let K and f_0 be as defined in the theorem statement. Let $\tilde{f}\colon K \to \mathbb{R}$ be defined for each simplex $\sigma \in K$ by $\tilde{f}(\sigma) := \max_{v\in\sigma} f_0(v)$. Let (H, T, C, m) and (H', T', C', m') be two GVFs that satisfy (i), (ii), and (iii).

Let $\sigma \in H$. By the forward direction of (ii), we know that σ is a left-right parent. By the backward direction of (ii), we know that $\sigma \in H'$. Thus, we have shown that $H \subseteq H'$. Repeating this argument by swapping the roles of H and H' gives us $H' \subseteq H$.

Since $\sigma \in H = H'$ and because (iii) holds, we have shown that σ is paired with $\rho(\sigma)$ in both matchings, and specifically $m(\rho(\sigma)) = \sigma = m'(\rho(\sigma))$. Since m and m' are bijections by (i), we also know that:

$$T = \{\tau \in K \mid \exists \sigma \in H \text{ s.t. } m(\rho(\sigma)) = \sigma\} = T'.$$

Thus, $T = T'$ and $m = m'$.

Finally, we conclude:

$$C = K \setminus (T \cup H) = K \setminus (T' \cup H') = C',$$

which means that (H, T, C, m) and (H', T', C', m') are the same GVF. Thus, we conclude that the gradient vector field satisfying (i), (ii), and (iii) is unique. □

Since ExtractRightChild and ExtractRaw both satisfy the hypothesis of Theorem 9, the outputs of the algorithms must be the same.

Theorem 10 (Algorithm Equivalence) *Let K be a simplicial complex and let $f_0\colon K_0 \to \mathbb{R}$ be an injective function. Then* ExtractRaw*(K, f_0) and* ExtractRightChild*(K, f_0) yield identical outputs.*

Proof. By Theorem 3 and Lemma 12, the output of ExtractRaw satisfies the properties in Theorem 9. By Lemma 8, Lemma 6, and Lemma 7, the output of ExtractRightChild satisfies the properties of Theorem 9. Then, by Theorem 9, ExtractRaw and ExtractRightChild are equivalent.

□

When we consider the runtime and space usage of ExtractRightChild, we find the following:

Theorem 11 (New Runtime) *Given a simplicial complex K (represented as a Hasse diagram), and an injective function $f_0 : K_0 \to \mathbb{R}$,* ExtractRightChild *computes a GVF consistent with f_0 in $O(dn)$ time and uses $O(n)$ space.*

Proof. First, line Line 1 decorates the Hasse diagram. By Lemma 4, the decoration takes $O(dn)$ time and $O(n)$ space. Lines 3-13, process each node of the decorated Hasse diagram. Each iteration of the loop is $O(1)$ in time and space because all required data was computed while decorating. As there are $n - n_0$ nodes to process, Lines 3-13 takes $O(n)$ time and uses $O(1)$ space. Finally, we iterate over the zero-simplices in $O(n_0)$ time.

The bottleneck of space and time usage of the algorithm is decorating the Hasse diagram, therefore, the algorithm takes $O(dn)$ time and $O(n)$ space. □

5 Discussion

In this paper, we identified properties of the Extract and ExtractRaw algorithms [13]. We used these properties to simplify ExtractRaw to the equivalent algorithm ExtractRightChild. Our simplification improves the runtime from $\Omega(n^2 \log n)$ to $O(dn)$.

There are several possible extensions of this work. The problem of finding tight bounds on the runtime of Extract is interesting and open. We plan to implement our approach on high dimensional data sets, and to further improve to the runtime. We intend to explore a cancellation algorithm that performs the same task as ExtractCancel, eliminating critical pairs with small persistence. Our conjectured cancellation algorithm iterates over critical simplices and applies ExtractRightChild.

Constructing Morse functions that do not require preassigned function values on the vertices is a related area of active research. The problem of finding a Morse function with a minimum number of critical simplices is NP-hard [12]. In [3], Bauer and Rathod show that for a simplicial complex of dimension $d \geq 3$ with n simplices, it is NP-hard to approximate a Morse matching with a minimum number of critical simplices within a factor of $O(n^{1-\epsilon})$, for any $\epsilon > 0$. The question is open for 2-dimensional simplicial complexes.

Acknowledgements This material is based upon work supported by the National Science Foundation under the following grants: CCF 1618605 & DMS 1854336 (BTF) and DBI 1661530 (DLM). Additionally, BH thanks the Montana State Undergraduate Scholars Program. All authors thank Nick Scoville for introducing us to KKM [13] and for his thoughtful discussions.

References

[1] U. Bauer. *Persistence in Discrete Morse Theory.* PhD thesis, Niedersächsische Staats-und Universitätsbibliothek Göttingen, 2011.

[2] U. Bauer, C. Lange, and M. Wardetzky. Optimal topological simplification of discrete functions on surfaces. *Discrete and Computational Geometry*, 47(2):347–377, 2012.

[3] U. Bauer and A. Rathod. Hardness of approximation for Morse matching. arXiv:1801.08380, 2018.

[4] L. Čomić and L. De Floriani. Dimension-independent simplification and refinement of Morse complexes. *Graphical Models*, 73(5):261–285, 2011.

[5] T. Dey, J. Wang, and Y. Wang. Graph reconstruction by discrete Morse theory. In *34th Symposium on Computational Geometry (SoCG)*, pages 31:1–31–13, 2018.

[6] H. Edelsbrunner and J. Harer. *Computational Topology: An Introduction*. American Mathematical Society, 2010.

[7] H. Edelsbrunner, J. Harer, and A. Zomorodian. Hierarchical Morse-Smale complexes for piecewise linear 2-manifolds. *Discrete and Computational Geometry*, 30(1):87–107, 2003.

[8] H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological persistence and simplification. *Discrete and Computational Geometry*, 28:511–533, 2002.

[9] R. Forman. Discrete Morse theory for cell complexes. *Advances in Mathematics*, 134:90–145, 1998.

[10] R. Forman. A user's guide to discrete Morse theory. *Séminaire Lotharingien de Combinatoire*, 42:Art. B48c, 35pp, 2002.

[11] P. Hersh. On optimizing discrete Morse functions. *Advances in Applied Math*, 35:294–322, 2005.

[12] M. Joswig and M. Pfetsch. Computing optimal Morse matchings. *SIAM Journal on Discrete Mathematics (SIDMA)*, 20(1):11–25, 2006.

[13] H. King, K. Knudson, and N. Mramor. Generating discrete Morse functions from point data. *Experimental Mathematics*, 14:435–444, 2005. MR2193806.

[14] K. Knudson. *Morse Theory: Smooth and Discrete*. World Scientific Publishing Company, 2015.

[15] T. Lewiner, H. Lopes, and G. Tavares. Toward optimality in discrete Morse theory. *Experimental Mathematics*, 12:271–285, 2003.

[16] J. Milnor. *Morse Theory*. Princeton University Press, Princeton, New Jersey, 1963.

[17] K. Mischaikow and V. Nanda. Morse theory for filtrations and efficient computation of persistent homology. *Discrete and Computational Geometry*, (50):330, 2013.

[18] R. Raz. On the complexity of matrix product. *SIAM Journal on Computing*, 32:1356–1369, 2003.

[19] N. Scoville. *Discrete Morse Theory*. American Mathematical Society, Providence, Rhode Island, 2019.

A Additional Details for Extract

To put our result in context, we now provide a glimpse into the inner workings of Extract, and reveal the underlying properties of ExtractRaw which give it an identical output to ExtractRightChild. We also provide a formal runtime analysis of ExtractRaw to verify that ExtractRightChild provides an improved time complexity.

A.1 Subroutines for Extract

In this section, we recall the algorithms proposed by KKM [13]. Note that we made some slight modifications to the presentation of KKM's initial description to improve readability. The modifications do not affect the asymptotic time or space used by the algorithm, although it does remove some redundant computation.

In particular, we modified the inputs to explicitly pass around a GVF so that the inputs of each algorithm are clear. We simplified notation and inlined the subroutine Cancel. From the previous modifications, we observed that the algorithm recomputes a gradient path that is currently in scope and so we simply unpack the path on Line 11 of Algorithm 4.

ExtractRaw computes the lower link of each vertex v in a simplicial complex, and assigns $v \in C$ if $\text{lowerlink}_K(v) = \emptyset$. If $\text{lowerlink}_K(v) \neq \emptyset$, its lower link is recursively inputted into ExtractRaw and this recursion continues until an empty lower link is reached. When the lower link is not empty, ExtractRaw assigns $v \in T$ and the smallest function valued vertex ω_0 in $\text{lowerlink}_K(v)$ is combined with v and added to H, carrying with this assignment a mapping m from $\omega_0 * v \in H$ to $v \in T$. As the recursion continues, higher dimensional simplices in the lower start of v are able to be assigned to both H and T based on combinations consistent with the assignments of the vertices and the original mappings of m. Higher dimensional critical cells are assigned similarly by combining the current vertex and each previously computed $\sigma \in C$ from the last recursion, until all simplices have been assigned.

Then, because ExtractRaw may have extraneous critical cells, Cancel works to reduce the number of critical cells by locating "redundant" gradient paths to a critical simplex and reversing them after the first pass by ExtractRaw, refining the output of Extract .

Algorithm 3 [13] ExtractRaw

Input: simp. complx. K, injective fcn. $f_0 : K_0 \to \mathbb{R}$
Output: a GVF consistent with f_0

1: $T \leftarrow \emptyset$, $H \leftarrow \emptyset$, $C \leftarrow \emptyset$, $m \leftarrow \emptyset$
2: **for all** $v \in K$ **do**
3: Let K' := the lower link of v.
4: **if** $K' = \emptyset$ **then**
5: Add v to C
6: **else**
7: Add v to T
8: $(T', H', C', m') \leftarrow$ Extract(K', f_0, ∞)
9: $w_0 \leftarrow arg\ \min_{w \in C'_0}\{f_0(w)\}$
10: Add $w_0 v$ to H
11: Define $m(w_0 * v) := v$
12: For each $\sigma \in C' \setminus \{w_0\}$, add $v * \sigma$ to C
13: **for all** $\sigma \in T'$ **do**
14: Add $v * \sigma$ to T
15: Add $v * m'(\sigma)$ to H
16: Define $m(v * \sigma) = v * m'(\sigma)$
17: **return** (H, T, C, m)

Let $p \in \mathbb{N}$, $\sigma \in K_p$ be a critical simplex. Let $\mathcal{G}^j_\sigma$

Algorithm 4 [13] ExtractCancel

Input: simplicial complex K, injective function $f_0 \colon K_0 \to \mathbb{R}$, $p \geq 0$, $j \in \mathbb{N}$, and GVF γ
Output: Gradient vector field on K

```
1: Let (H, T, C, m) be the four components of γ
2: for all σ ∈ C_j do
3:     s ← max_{v∈σ} f_0(v)
4:     S ← {Γ | Γ ∈ G^j_σ, s − max_{w∈Γ_L} f_0(w) < p}
5:     for all Γ ∈ S do
6:         m_Γ ← ∞
7:         if Γ_L ≠ Γ'_L for any other Γ' ∈ S then
8:             m_Γ ← max_{w∈Γ_L} f_0(w)
9:     Γ* ← arg min_{Γ∈S}{m_Γ}
10:    if m_{Γ*} ≠ ∞ then
11:        {σ_1, τ_1, ··· , σ_k, τ_k} ← Γ*
12:        Remove τ_k, σ_1 from C
13:        Add τ_k to T; Add σ_1 to H
14:        Add (τ_k, σ_k) to m
15:        for i = 1, ..., k − 1 do
16:            Remove (τ_i, σ_{i+1}) from m
17:            Add (τ_i, σ_i) to m
18: return (H, T, C, m)
```

denote the set of all nontrival gradient path starting at $\sigma \in C_j$ and ending in C_{j-1}.

A.2 Analysis of ExtractRaw

In this appendix, we provide the analysis Algorithm 1 from Section 3. In what follows, let K be a simplicial complex and let $f_0 \colon K_0 \to \mathbb{R}$ be an injective function.

Lemma 12 (Raw Heads are Parents) *Let (H, T, C, m) be the output of $\textsc{ExtractRaw}(K, f_0)$. Every simplex in H is a left-right parent. Furthermore, for all $\sigma \in H$, $m(\rho(\sigma)) = \sigma$.*

Proof. Let $\sigma \in H$. We show that σ is a left-right parent by induction on the dimension of K. When $\dim(K) = 1$, σ is an edge, and $\sigma = m(\tau)$ for some vertex $\tau \in T$. In Line 10 of Algorithm 3 σ is defined as $m(\tau) = [w_0, \tau]$ where $w_0 \in C'_0$ so that $f_0(w_0)$ is smallest. So, σ is a left-right parent. Furthermore, $m(\rho(\sigma)) = m(\rho([w_0, \tau])) = m(\tau) = \sigma$.

Suppose every $\sigma \in H$ is a left-right parent when $\dim(K) \leq d$ and consider $\dim(K) = d + 1$. If σ is a $(d+1)-$simplex, $\sigma = m(\tau)$ is defined in Line 15 of Algorithm 3, when a vertex v is selected in Line 2 of Algorithm 3. We extend the GVF on the $\text{lowerlink}_K(v)$ to include the lower star of v. We have $m(\tau) = v * m'(\alpha)$ where $\alpha = [v_1, \ldots, v_d] \in T'$, $m'(\alpha) = [v_0, v_1 \ldots v_d] \in H'$. Since α and $m'(\alpha)$ are in the $\text{lowerlink}_K(v)$ we have $f(v_i) < f(v)$ for $0 \leq i \leq d$. Then $\tau = v * \alpha = [v_1, \ldots, v_d, v]$ and $\sigma = m(\tau) = [v_0, v_1, \ldots, v_d, v]$.

By the induction hypothesis $m'(\alpha) \in H'$ is a left-right parent. If σ is not a left-right parent, we can remove v from σ and τ and contradict that $m'(\alpha) \in H'$.

Furthermore, $m(\rho(\sigma)) = m(\rho(m(\tau))) = m(\rho([v_0, v_1, \ldots, v_d, v])) = m([v_1, \ldots, v_d, v]) = m(\tau) = \sigma$. This proves the claim. □

Lemma 13 (Raw Parents are Heads) *Let (H, T, C, m) be the output of $\textsc{ExtractRaw}(K, f_0)$. Let $\sigma \in K$. If σ is a left-right parent, then $\sigma \in H$.*

Proof. We show if $\sigma \in T \cup C$ then σ is not a left-right parent. First, suppose $\sigma \in T$. We use induction on $\dim(K)$ to show σ is not a left-right parent. For the base case, $\dim(K) = 1, \sigma$ is a vertex and can not be a left-right parent.

Suppose $\sigma \in T$ is not a left-right parent when $\dim(K) \leq d$ and consider $\dim(K) = d + 1$. Then σ is added to T in Line 14 of Algorithm 3 when a vertex v is selected in Line 2. As in Lemma 12, write $\sigma = v * \alpha = [v_1, \ldots, v_d, v]$ for some $\alpha \in T'$.

By the induction hypothesis α is not a left-right parent, thus $\ell \circ \rho(\alpha) \neq \alpha$, and there exists a vertex v_{-1} such that $\ell(\rho(\alpha)) = [v_{-1}, v_2, v_3 \ldots, v_d]$ where $f(v_{-1}) < f(v_1)$. We have $\ell \circ \rho(\sigma) = \ell \circ \rho([v_1, v_2, \ldots, v_d, v]) = \ell([v_2, v_3 \ldots, v_d, v]) = [v_{-1}, v_2, \ldots, v_d, v] \neq \sigma$.

Now, suppose $\sigma \in C$. There are two places where elements are added to C, Line 5 of Algorithm 3 and Line 12. In Line 5 c is a vertex and can not be a left-right parent.

In Line 12 c is defined as $c = v * \alpha$ for some $\alpha = [v_0, v_1, \ldots, v_d] \in C' \backslash w_0$ where $w_0 \in C'_0$ so that $f_0(w_0)$ is smallest. Now $\ell \circ g(c) = \ell \circ g([v_0, v_1, \ldots, v_d, v]) = \ell([v_1, \ldots, v_d, v]) = [w_0, v_1, \ldots, v_d, v] \neq c$. We have shown that if σ is a left-right parent, then $\sigma \in H$.

□

We summarize the properties of ExtractRaw in the following theorem.

Theorem 3 (Properies of ExtractRaw) *Let K be a simplicial complex, let $f_0 \colon K_0 \to \mathbb{R}$ be an injective function, and suppose (H, T, C, m) is the output of $\textsc{ExtractRaw}(K, f_0)$. Let $\varepsilon > 0$. Then, there exists a discrete Morse function $f \colon K \to \mathbb{R}$ such that the following hold:*

(i) (H, T, C, m) is a GVF consistent with f_0.

(ii) Let $\sigma \in K$. Then, $\sigma \in H$ if and only if σ is a left-right parent.

(iii) For all $\sigma \in H$, $m(\rho(\sigma)) = \sigma$.

(iv) The runtime of ExtractRaw is $\Omega(n^2 \log n)$.

Proof. (i) is proven in Theorem 3.1 of [13]. By Lemma 12 and Lemma 13, we conclude (ii). Also by Lemma 12 we can guarantee (iii).

To show (iv), we observe that the worst-case runtime for a single execution of Line 8 of Algorithm 3 happens when the lower link of v is of size $\Theta(n/2)$. Computing the optimal pairings that ExtractRaw returns is at least as hard as computing the homology of K', which is of the time complexity of matrix multiplication. By [18], we know that the runtime of ExtractRaw is lower-bounded by $\Omega(n^2 \log n)$. □