Discrete Morse Theory Algorithms

ScuDo
Scuola di Dottorato ~ Doctoral School
WHAT YOU ARE, TAKES YOU FAR

Doctoral Dissertation
Doctoral Program in Mathematics For Engineering Sciences (28^{th}cycle)

Discrete Morse Theory
Algorithms

By

Soroosh Nazem

******

Supervisor:
Prof. Francesco Vaccarino

Doctoral Examination Committee:
Dott. Andrea Villa, Referee, CNR, Pisa
Prof. Federica Galluzzi, Referee, Università di Torino
Prof. Alberto Albano, Università di Torino
Prof. Paolo Brandimarte, Politecnico di Torino
Prof. Paolo Cermelli, Università di Torino

Politecnico di Torino
2017

Declaration

I hereby declare that, the contents and organization of this dissertation constitute my own original work and does not compromise in any way the rights of third parties, including those relating to the security of personal data.

Soroosh Nazem
2017

* This dissertation is presented in partial fulfillment of the requirements for **Ph.D. degree** in the Graduate School of Politecnico di Torino (ScuDo).

Abstract

Discrete Morse Theory (DMT) is the discrete version of Morse Theory (see [1]) and has been introduced by Robin Forman in [2]. Discrete Morse Theory provides a powerful tool for the analysis of topological spaces. The main focus of DMT, like its predecessor Morse theory, is based on finding critical points and constructing a simpler topological space which is homotopy equivalent (see C.0.2) to the original topological space.

In this thesis, we will introduce discrete Morse function and explain how we can find it over a simplicial complex. We will explain how finding a discrete Morse function can be converted to a combinatorial problem which is called a discrete Morse matching. Then we will explain the algorithms for finding discrete Morse matchings over simplicial complexes. Here, we propose a parallel algorithm for discrete Morse matching computation. Then we show how to construct a Morse complex and compute its homology groups. In section 4.2, we propose an algorithm to compute homology groups based on parallel matching. Finally, we will see the results on some data sets. We have used the library of triangulations in [3]. We will see the effect of parallelization on elapsed time of the algorithm for different simplicial complexes.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Discrete Morse Theory (DMT) is the discrete version of Morse Theory (see [1]) and has been introduced by Robin Forman in [2]. Discrete Morse Theory provides a powerful tool for the analysis of topological spaces. The main focus of DMT, like its predecessor Morse theory, is based on finding critical points and constructing a simpler topological space which is homotopy equivalent (see C.0.2) to the original topological space.

In this thesis, in chapter 2, we will give a short introduction to discrete Morse theory, we will introduce discrete Morse function and explain how we can find it over a simplicial complex, then we briefly explain how a Morse complex which is homotopy equivalent to the simplicial complex can be constructed. We will show finding a discrete Morse function can be converted to a combinatorial problem which is called discrete Morse matching.

In chapter 3, we will explain the algorithms for finding discrete Morse functions. Here we introduce a parallel algorithm for discrete Morse matching computation.

In chapter 4, we show how we can construct a Morse complex and compute its homology groups. In section 4.2, we introduce an algorithm that computes the Morse matching and constructs the Morse complex concurrently.

In chapter 5, we will see the results on some data sets. We have used the library of triangulations in [3]. We will see the effect of parallelization on elapsed time of the algorithm for different simplicial complexes.

Finally, we will suggest the possible progresses that can be done beyond this thesis.

Chapter 2

Discrete Morse Theory

Discrete Morse Theory(DMT) provides us a powerful tool for calculating homology groups of a simplicial complex. This theory was developed by Robin Forman ([2]) in the 1990s as a combinatorial version of Morse theory.

In DMT the simplices are labeled as either *critical* or *matched*. Then a new complex which is called *Morse Complex* is constructed by just critical simplices. The homology groups of this new complex are isomorphic to the homology groups of original simplicial complex (see [4]). To construct a Morse complex, after finding critical and matched simplices, we need to find boundary operator that in this situation we call *Morse boundary operator* and we denote it by ∂_*^{MM}. To calculate these operators, we need to calculate the connectivity among critical simplices and this is done by calculating *V-paths* among critical simplices. V-paths will be explained in this chapter.

Definition 2.0.1 (Discrete Morse Function)**.** *Let K be a simplicial complex. A function $f : K \to \mathbb{R}$ is called a discrete Morse function, if for every simplex τ it holds that:*

1. $|\{\sigma \rhd \tau | f(\sigma) \leq f(\tau)\}| \leq 1$
2. $|\{\gamma \lhd \tau | f(\gamma) \geq f(\tau)\}| \leq 1$

where $|\Delta|$ denotes the cardinality of the set Δ.

This definition says that the value given to a simplex has to be greater than all of its proper faces, It is just allowed to be equal or smaller than one of them. This value has to be smaller than all of its proper cofaces, except one.

Proposition 2.0.1. $|\{\sigma \rhd \tau | f(\sigma) \leq f(\tau)\}| + |\{\gamma \lhd \tau | f(\gamma) \geq f(\tau)\}| \leq 1$

In another word, this proposition says at most one of the constraints of conditions (1) and (2) in Discrete Morse function can be equal to 1.

Proof. Assume both $|\{\sigma \rhd \tau | f(\sigma) \leq f(\tau)\}|$ and $|\{\gamma \lhd \tau | f(\gamma) \geq f(\tau)\}|$ are equal to 1. So there is a simplex $\sigma \rhd \tau$ such that $f(\sigma) \leq f(\tau)$ and there is a simplex $\gamma \lhd \tau$ such that $f(\gamma) \geq f(\tau)$. So $f(\sigma) \leq f(\tau) \leq f(\gamma)$. Consider another proper face of σ, τ' that contains γ. Since already $f(\sigma) \leq f(\tau)$, so $f(\sigma) > f(\tau')$. Therefor, $f(\tau') < f(\gamma)$; this is a contradiction, because the $f(\gamma)$ can be equal or greater than at most one of its proper cofaces, while right now it has become two, τ and τ'. □

Definition 2.0.2 (Critical)**.** *Let $f : K \to \mathbb{R}$ be a discrete Morse function. A k-simplex τ is called critical if:*

1. $|\{\sigma \rhd \tau | f(\sigma) \leq f(\tau)\}| = 0$
2. $|\{\sigma \lhd \tau | f(\sigma) \geq f(\tau)\}| = 0$

Otherwise, it is called regular or matched.

As an example consider the function $f(\sigma) = \dim \sigma$. In this function all simplices are critical, because for every simplex, the given value is bigger than the given values of proper faces and smaller than the given values of proper cofaces. In figure 2.1, we have two Morse functions: in one of them all simplices are critical and in the other there is just one 0-simplex as a critical cell. In [5], a discrete Morse function is defined on a torus. At first let us see a theorem which shows the importance of discrete Morse theory.

Definition 2.0.3. *For a discrete Morse function on an n-dimensional complex, the Discrete Morse Vector is a vector $C = (c_0, c_1, \cdots, c_n)$ where c_i represents the number of $i-$dimensional critical simplices.*

In figure 2.1, the Discrete Morse Vector for the values in the left hand side is $C_1 = (3, 3, 1)$, because all simplices are critical, while for the right hand side is $C_2 = (1, 0, 0)$.

Theorem 2.0.1. *Let K be a simplicial complex with a discrete Morse function $f : K \longrightarrow \mathbb{R}$ and Discrete Morse Vector $(c_0, c_1, \cdots, c_n)$. Then K is homotopy equivalent to a cell complex whose number of $i-$dimensional cells is c_i, for $i = 0, \ldots, n$. (See A.0.2)*

Proof. See [6] □

Afterward, we will explain how the cell complex mentioned in Theorem 2.0.1 will be constructed. We call it the **Morse complex**. We have the following theorem.

Theorem 2.0.2 (Morse Inequalities)**.** *Let K be a simplicial complex of dimension n with Betti numbers β_k, $k = 0, 1, \cdots, n$, f is a discrete Morse function on K with Discrete Morse Vector $(c_0, c_1, \cdots, c_n)$. Then*

1. $c_k \geq \beta_k$ *for all k, (**weak Morse inequality**)*
2. $\sum_{i=0}^{k}(-1)^{k-i}c_i \geq \sum_{i=0}^{k}(-1)^{k-i}\beta_i$ *,for all k, (**strong Morse inequality**)*
3. $\sum_{i=0}^{n}(-1)^{n-i}c_i = \sum_{i=0}^{n}(-1)^{n-i}\beta_i = \mathcal{X}(K)$. *(**Poincaré -Hopf equality**)*

*where $\mathcal{X}(K)$ is the **Euler characteristic** (see C.0.4) number of K.*

Proof. See [7](pag. 29). □

Definition 2.0.4 (Perfect Morse Function)**.** *A discrete Morse function is called perfect, if the number of critical cells in each dimension is equal to Betti number.*

For a perfect discrete Morse function, the inequalities in theorem 2.0.2 will be equalities.

Definition 2.0.5 (Optimal Morse Function)**.** *A discrete Morse function f on an n-dimensional simplicial complex K, with discrete Morse vector $C_f = (c_{f,0}, c_{f,1}, \cdots, c_{f,n})$ is optimal, if for any discrete Morse function g on K with discrete Morse vector $C_g = (c_{g,0}, c_{g,1}, \cdots, c_{g,n})$:*

$$c_{f,i} \leq c_{g,i}, \ \forall i = 0, \cdots, n$$

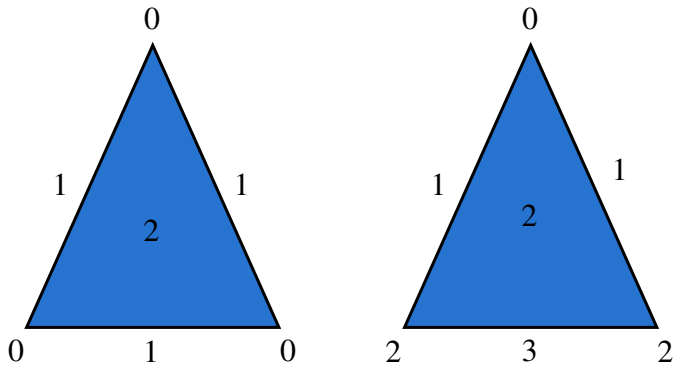


Fig. 2.1 $f(\sigma) = dim(\sigma)$ (left), Morse funtion with just one critical cell(right)

C_f is called the optimal DMV of K.

Definition 2.0.6 (Matching)**.** *A matching in a simplicial complex K is a pair of (τ, σ) such that $\tau \lhd \sigma$*

Definition 2.0.7 (Discrete Vector Field)**.** *A discrete vector field V on K is a collection of matchings (τ, σ) such that each simplex is present in at most one matching.*

Consider a discrete Morse function $f : K \to \mathbb{R}$. We are going to find a discrete vector field V, induced by f. In K consider a simplex τ. If τ is critical with respect to f, then we do not put it in any matching.

If τ is regular, there is just one either 1-codimensional face or 1-codimensional coface σ of τ such that $f(\sigma) \geq f(\tau)$ or $f(\sigma) \geq f(\tau)$, respectively.

If $\tau \lhd \sigma$, then we add (τ, σ) and if $\tau \rhd \sigma$, then we add (σ, τ) to V. In this case every simplex is either absent in V or it is present in at most one matching. So we have created a discrete vector field.

Definition 2.0.8 (gradient vector field)**.** *Suppose $f : K \to \mathbb{R}$ is a discrete Morse function on a simplicial complex K. The discrete vector field which is obtained from a discrete Morse function f as above, is called the **gradient vector field** of f and we denote it by $-\bigtriangledown f$*

Example 2.0.1. *Consider the function $f(\sigma) = dim(\sigma)$. As we said before, in this function, all simplices are critical. Therefore, the gradient vector field of f is an empty set, $-\bigtriangledown f = \varnothing$.*

Example 2.0.2. *Let K be the simplicial complex depicted in figure 2.2 at page* 11. *Let $g : K \to \mathbb{R}$ be given by:*

$g([0]) = 0\,, g([1]) = g([2]) = g([012]) = 2\,, g([01]) = g([02]) = 1\,, g([12]) = g([13]) = 3\,, g([3]) = 4\,, g([23]) = 5$

As you see here there are two critical simplices: [0] *and* [23] *and the others are matched. Here the gradient vector of g is $-\bigtriangledown g = \{([1],[01]),([2],[02]),([12],[012]),([3],[13])\}$.*

So far, we have understood that a discrete Morse function gives us implicitly a discrete vector field, which we call it the gradient vector field, but not all discrete vector fields are the gradient vector field of a discrete Morse function. A simple counterexample is the following:

Example 2.0.3. *Consider a triangle with three vertices* [0], [1], [2]. *Consider this discrete vector field $V = \{([0],[01]),([1],[12]),([2],[02])\}$. Consider* ([0], [01])*, in the associated discrete Morse function f, we should have $f([0]) \geq f([01])$. Moreover, by the definition of discrete Morse function, $f([01]) > f([1])$. If we repeat the same reasoning for two other vector in V, we will have $f([1]) \geq f([12]) > f[2]$ and $f([2]) \geq f([02]) > f[0]$. If we combine these relations we have:*

$$f([0]) \geq f([01]) > f([1]) \geq f([12]) > f([2]) \geq f([02]) > f([0])$$

So $f([0]) > f([0])$ which is a contradiction. Therefore, V is not associated to any Discrete Morse function

We want to characterize which discrete vector fields are gradient vector fields of a discrete Morse function.

Definition 2.0.9 (V Path on a discrete vector field)**.** *Let V be a discrete vector field. A V-path is a sequence of simplices*

$$\sigma_0, \tau_1, \sigma_1, \tau_2, \sigma_2 \cdots, \sigma_{r-1}, \tau_r$$

such that for each $i = 1, 2, \cdots, r-1$, $(\tau_i, \sigma_i) \in V$ and $\sigma_i \rhd \tau_{i+1} \neq \tau_i$. Such a path is called nontrivial closed path if $r \geq 0$ and $\tau_r \lhd \sigma_0$.

Example 2.0.4. *Consider the gradient vector field $-\bigtriangledown g$ in example 2.0.2 the following sequence:*

$$[23],\ [3],\ [13],\ [1],\ [01],\ [0]$$

is a path induced by discrete Morse function g and as you see, this path is not closed.

Right now consider the discrete vector field $V = \{([0],[01]),([1],[12]),([2],[02])\}$, the V-path is:

$$[0],\ [01],\ [1],\ [12],\ [2],\ [02]$$

this V-path is closed, because $[0] \lhd [02]$

Now, we state the following theorem that makes a connection between discrete Morse function and the V-path

Theorem 2.0.3. *Suppose V is the gradient vector field of discrete Morse function f. Then a sequence of simplices is a V-path if and only if $\tau_i^k \lhd \sigma_i^{k+1}$ for $i = 0, 1, \cdots, r$ and $f(\tau_0^k) \geq f(\sigma_0^{k+1}) > f(\tau_1^k) \geq f(\sigma_1^{k+1}) > \cdots \geq f(\sigma_r^{k+1}) > f(\tau_{r+1}^k)$.*

Proof. See [8] on page 94. □

Example 2.0.5. *In example 2.0.4, look at the V-path created by $-\bigtriangledown g$ and then substitute the elements of V-path by their values taken from g, we have:*

$$g([0]) \geq g([01]) \geq g([1]) \geq g([12]) \geq g([2]) \geq g([02])$$

We are now in the position to present this important theorem that makes a mutual relation between discrete vector field and discrete Morse function.

Theorem 2.0.4. *A discrete vector field V is the gradient vector field of a discrete Morse function, if and only if there is no nontrivial closed V-path.*

Proof. See [2] □

So, by this theorem instead of looking for a discrete Morse function we can look for a discrete vector field with no closed V-path. This issue converts our problem to a combinatorial problem. Before keep going forward, let us clarify the point by an example.

Example 2.0.6. *Consider the simplicial complex in figure 2.2. Now, look at the two following discrete vector fields over this complex:*

$$V_1 = \{\{[0],[01]\},\{[1],[12]\},\{[3],[13]\}\}$$
$$V_2 = \{\{[0],[02]\},\{[1],[01]\},\{[2],[12]\}\}$$

For V_1, we can not find any closed path. So with respect to theorem 2.0.4, it is a gradient vector field of a discrete Morse function. An example $f : X \to \mathbb{R}$ of this Morse function induces by V_1 is :

$$f([0]) = 2,\ f([01]) = 2,\ f([1]) = 1,\ f([12]) = 1,\ f([3]) = 2,\ f([13]) = 2$$
$$f([2]) = 0,\ f([02]) = 3,\ f([23]) = 3,\ f([012]) = 4$$

On another hand, look at V_2. there is a closed path in these discrete vector field:

$$[0],[02],[2],[12],[1],[01],[0]$$

With respect to the theorem 2.0.4, no discrete Morse function can be defined over V_2

2.1 Morse Matching

Herein, we introduce Hasse diagram which is helpful for a different representation of a simplicial complex.

Definition 2.1.1 (Hasse diagram)**.** *The Hasse diagram G_K of a simplicial complex K, is a directed graph such that every node of this graph corresponds to a simplex of K and $u, v \in V(G_K)$ are connected by an edge which is directed from u to v, if and only if the simplex corresponding to u is a 1-codimensional face of the simplex corresponding to v.*

A simplicial complex and its Hasse diagram are presented in figure 2.2. Each directed edge goes from coface to one of its faces. The advantage of Hasse diagram is that it makes the matching procedure for finding a discrete vector field more intuitive. Look at figure 2.3. In the left, you see the simplicial complex, and each arrow shows the two simplices which are matched. This arrow goes from a simplex to its matched coface. When a face-coface pair get matched and considered in discrete

vector field, in Hasse diagram, we reverse the direction of arrow that connects these two simplices. You see this modified Hasse diagram in figure 2.3. A V-path in the gradient vector field is equivalent to a path in Hasse diagram that alternates among the nodes.

Look at figure 2.3. We can simply discover the V-paths. For example:

[23], [3], [13], [1], [01], [0]

Meanwhile, since the discrete vector field should not have any closed path, equivalently in Hasse diagram we should not have any loop. Now, we have the following theorem.

Theorem 2.1.1. *A discrete vector field V is the gradient vector field of a discrete Morse function on K, if and only if modified Hasse diagram has no directed loops*

Proof. See [8] page 102. □

Therefore, according to theorem 2.0.4, our problem will be converted to a combinatorial problem instead of functional problem and it can be easier in computational point of view. In the next chapter we will explain some matching algorithms for this purpose, meanwhile we introduce a new method for parallelizing the matching. But here at first, we explain how a Morse complex can be built after matching.

Definition 2.1.2 (Partial Matching)**.** *Let G be a directed graph. A* ***partial matching*** $\mathscr{M}$ *in G is a subset of edges of G such that each node is adjacent to at most one edge in* $\mathscr{M}$*. The partial matching* $\mathscr{M}$ *is* ***acyclic*** *if upon reversing the orientation of the edges in* $\mathscr{M}$*, the resulting graph has no loops.*

Definition 2.1.3 (Morse Matching)**.** *An acyclic partial matching* $\mathscr{M}$ *is called* ***Morse Matching****.*

So, a discrete vector field on a simplicial complex is a partial matching and a gradient vector field is a Morse matching. In figure 2.3, you see a gradient vector field on a simplicial complex and on its modified Hasse diagram, you see the Morse matching.

Definition 2.1.4 (Perfect Morse Matching)**.** *A Morse Matching* $\mathscr{M}$ *is called a* ***Perfect Morse Matching****, if it is correspondent to a perfect Morse function.*

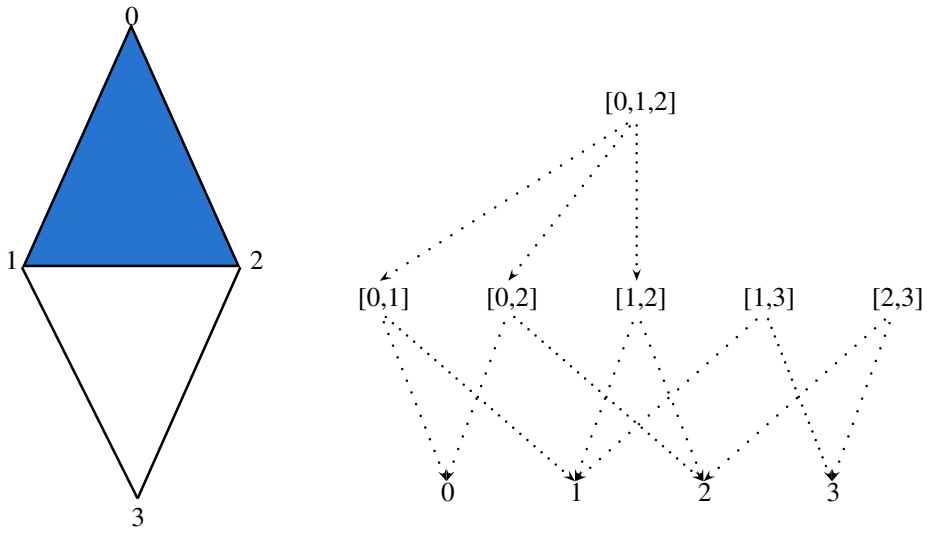


Fig. 2.2 simplicial complex(left) and Hasse Diagram(right)

As an example, the Morse matching is figure 2.3, is a perfect Morse matching, because the number of unmatched (in this case, critical) simplices is equal to Betti numbers.

2.2 Morse Complex

After matching and having a gradient vector field, we build a new complex called **Morse Complex**. We emphasize that we are working with $\mathbb{Z}_2$ coefficients.

Definition 2.2.1 (Morse Complex)**.** *Consider the discrete Morse function $f : K \to \mathbb{R}$ and let $Crit_k(f)$ be the set of k dimensional critical simplices in the simplicial complex K. The chain group $C_k(f)$ is a $\mathbb{Z}_2$-module generated by $Crit_k(f)$:*

$$C_k(f) = \{\sum_k \alpha_k \tau_k,\ \tau_k \in Crit_k(f),\ \alpha_k \in \mathbb{Z}_2\},\ \forall k = 0, 1, \cdots, dim(K)$$

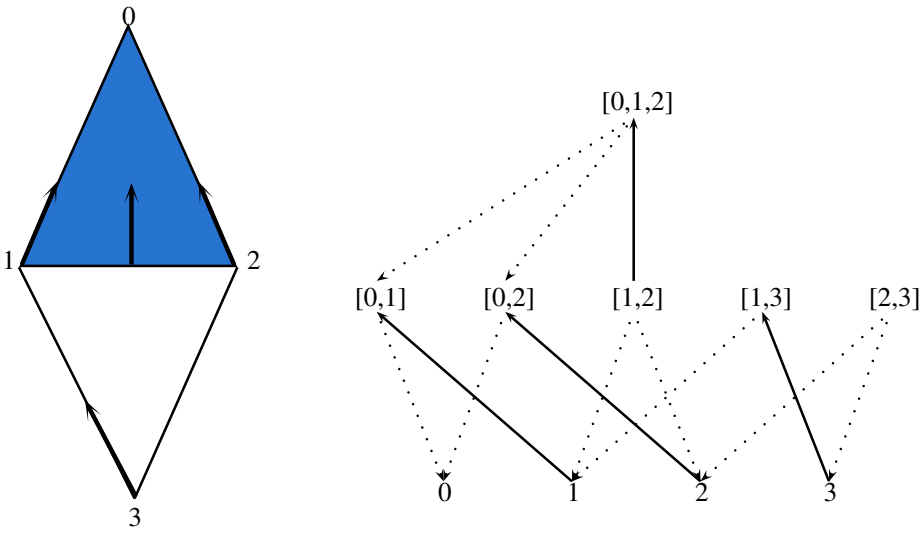


Fig. 2.3 Mathched simplicial complex(left) and Modified Hasse Diagram(right)

and the boundary operator $\partial_k : C_k(f) \longrightarrow C_{k-1}(f)$ *is defined by counting the number of V-paths between* $\sigma \in Crit_k(f)$ *and* $\tau \in Crit_{k-1}(f)$*:*

$$\partial_k(\sigma) = \sum_{\tau \in Crit_{k-1}(f)} |\mathscr{P}(\sigma, \tau)| \, mod \, 2\tau$$

where $\mathscr{P}(\sigma, \tau)$ *is the set of V-paths between* σ *and* τ*.*

For a more general definition of Morse complex and to understand why we can build a complex by this approach, see [9]. In order to clarify the definition of Morse complex, we give the following example.

Example 2.2.1. *Consider the simplicial complex K in figure 2.2. The boundary operator matrices are*

$$\partial_1 = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}, \quad \partial_2 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

After matching, as shown in figure 2.3, just [0] *and* [23] *are left as criticals and others are matched. So the Morse complex extracted from this Morse matching has one 0-simplex, one 1-simplex and no 2-simplex. Now we need to count the number*

of V-paths between [0] *and* [23]*:*

$$\mathscr{P}([23],[0]) = \{([23],[3],[13],[1],[01],[0]),([23],[2],[02],[0])\}$$

So $|\mathscr{P}([23],[0])| = 2$ *and the boundary operator matrices will be*

$$\partial_1^M = \begin{bmatrix} 0 \end{bmatrix}$$

and as you see the homology groups of this complex are isomorphic to the original complex.

In the next chapter, some algorithms for counting the number of V-paths will be explained. Right now, in order to mention a point, let us see another example.

Example 2.2.2. *Consider an empty tetrahedron. Its Hasse diagram is depicted in figure 2.4. In figure 2.5, you see how we matched the simplices to build a discrete vector field* V*:*

$$V = \{([1],[01]),([2],[02]),([3],[03]),([12],[012]),([13],[013]),([23],[023])\}$$

As you see, [0] *and* [123] *are critical simplices. Since we have no 1-simplex as critical, so we can not have any V-path, and so we have no boundary operator like* ∂_1^M *and* ∂_2^M*. So in this case we can simply just say that* $\beta_0 = 1$*,* $\beta_1 = 0$*,* $\beta_2 = 1$ *and we do not need to calculate any V-path.*

2.3 Some Notes on Perfect Morse Function

As explained before, in perfect Morse functions, the number of critical simplices in each dimension is equal to the Betti number of that dimension. Unfortunately, we can not find perfect Morse functions for all simplicial complexes and there are some limitations. For example, torsion is considered as an obstruction for a perfect Morse matching. Some other parameters like being contractible are also important.

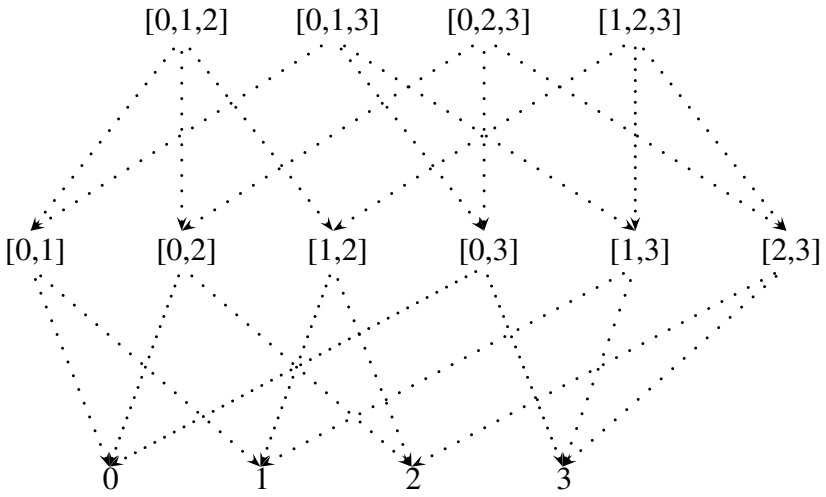


Fig. 2.4 Hasse Diagram of an Empty Tetrahedron

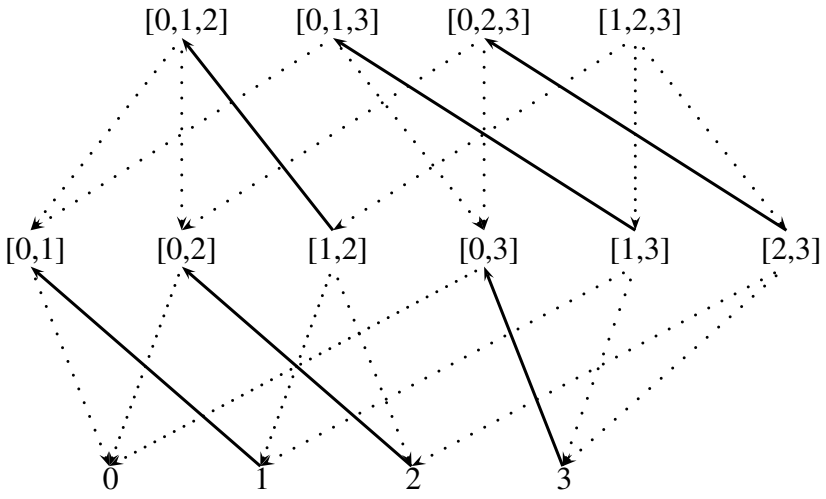


Fig. 2.5 Hasse Diagram of Tetrahedron after matching

One of the well known simplicial complexes that does not accept a perfect Morse function is dunce hat. In [10], [11] and [12], there are some explanations about perfect Morse functions and why some simplicial complexes do not accept perfect Morse functions. Here, we just give a proposition.

Proposition 2.3.1. *Every 1-dimensional simplicial complex has a perfect Morse function.*

Proof. See [12] page 3. □

All algorithms do not guarantee a perfect Morse function for a 1-dimensional simplicial complex, but some do. We will introduce one of them. The advantage of this proposition is that if we use an algorithm that guarantees perfect Morse function, then we are sure that $(c_0, c_1) = (\beta_0, \beta_1)$ and so for calculating the homology groups and Betti numbers we do not need to calculate V-paths, unless we need the generators of homology groups.

Chapter 3

Discrete Morse Matching Computation

In this chapter, we explain two major strategies for computing discrete Morse matching. Referring to theorem 2.0.4, finding discrete Morse function can be translated to a discrete gradient Vector field and by looking at Hasse diagram, we need to find an acyclic matching such that each node is present in at most one matching. In [13], it is proven that finding an optimal Morse matching is an **NP**-hard problem. In any case,the first method will be based on elementary collapses and the second method will be based on elementary coreduction. In [14], another method based on Fourier transform has been introduced.

3.1 Elementary Collapses Method

Here at first, we need to define some terminologies.

Definition 3.1.1 (Free Face)**.** *A k-simplex is called a **Free Face**, if it is a 1-codimensional face of just one (k+1)-simplex.*

Free face is actually a generalization of leaves in graph theory. For example, in the simplicial complex shown in figure 2.2, the 1-simplices [01], [02] and [12] are free faces of [012].

Definition 3.1.2 (Elementary Collapse)**.** *An elementary collapse is the deletion of a free face together with its coface.*

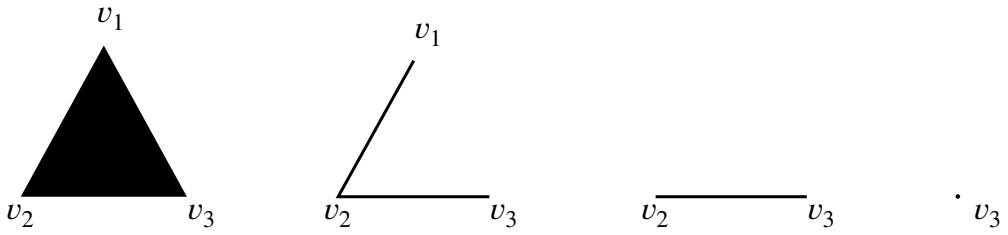


Fig. 3.1 Example of a sequence of elementary collapses

Definition 3.1.3 (collapsible complex)**.** *A simplicial complex is called collapsible, if it has a sequence of collapses leading to a point.*

In figure 3.1, you see a sequence of elementary collapses; we start with a triangle and end up to a single node. As you know,the Betti numbers of a fulfilled triangle and a node are equal. So, we can claim that elementary collapses do not influence the homology groups of a simplicial complex. We formalize this claim in the following theorem.

Theorem 3.1.1. *A sequence of elementary collapses yields to a homotopy equivalence.*

Proof. See [15], page 93. □

In another word, this theorem says that if we have a sequence of elementary collapses, all the complexes during this procedure are homotopy equivalent and so their homology groups are isomorphic. In figure 3.1, all the complexes have isomorphic homology groups.

The algorithm based on elementary collapses works in this way: as long as there is a free face, we do an elementary collapse. When there is no free face, we remove one of the facets from the complex and mark it as critical. We continue this approach till achieving a single node. The final single node is also marked as critical. In figure 3.2, you see that in the triangle since there no free face, we remove one of the edges as critical and then we continue the collapsing procedure. So, $c_0 = 1$, $c_1 = 1$, where c_i denotes the number of i-dimensional critical points.

Lemma 3.1.1. *The algorithm based on elementary collapses gives a discrete Morse matching of the modified Hasse diagram of a simplicial complex.*

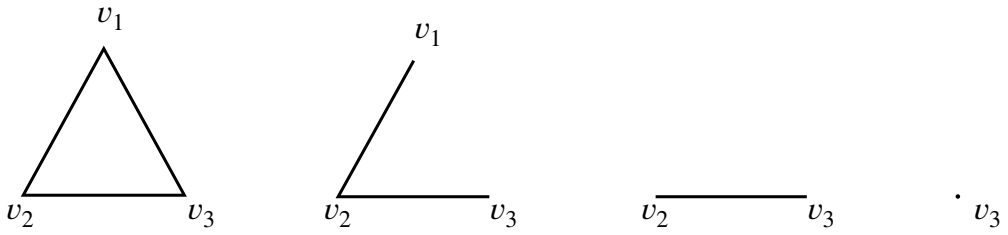


Fig. 3.2 Example of a complex that doesn't have any free face at beginning and then by deleting one edge as critical cell, we will have free faces

Proof. Consider the Hasse diagram of a simplicial complex. An elementary collapse is the deletion of a pair of simplices (τ, σ) where $\tau \lhd \sigma$ and τ is a free face. If we perform a sequence of successive elementary collapses we obtain, therefore, a partial matching by the definition of free face.

Now, in order to show that this is a Morse matching, we have to show that it is acyclic. We prove this by absurd. Let us assume that we have the following sequence of matchings

$$(\tau_1, \sigma_1), (\tau_2, \sigma_2), \cdots (\tau_n, \sigma_n), \text{ with } \tau_i \lhd \sigma_{i-1} \text{ for } i = 2, \dots, n$$

inducing a path in the modified Hasse diagram. Assume that $\tau_1 \lhd \sigma_n$ to obtain a cycle. Observe that, when we matched τ_1 and σ_1, τ_1 was a free face and this means that its other cofaces had been earlier removed as critical or as matched. But the matching (τ_n, σ_n) is successive to (τ_1, σ_1), thus the only possibility for σ_n to be a coface of both 1 and τ_n is that it was critical which is a contradiction. □

Definition 3.1.4 (Matching Vectors)**.** *Consider k-simplices in a simplicial complex* K *and a Morse matching on* K*. A* ***matching vector*** $\mathcal{N}_k$ *for k-simplices is a vector whose dimension is the number of k-simplices in* K*, and its i'th component is:*

$$\mathcal{N}_k(i) = \begin{cases} -j & \textit{matched with j-th } (k+1)\textit{-simplex} \\ j & \textit{matched with j-th } (k-1)\textit{-simplex} \\ 0 & \textit{critical} \end{cases} \quad \forall k = 0, \cdots, dim\,(K)$$

Matching vectors are needed for calculating V-paths and Morse boundary operators for a Morse matching. In the following, we give some examples.

Algorithm 1: Morse Matching Based on Elementary Collapse

Input: boundary matrices ∂_* of simplicial complex K
Output: matching vectors $\mathcal{M}_*$

```
0 :  Initialize M_k = 0⃗  ∀k = 0, ··· , dim(K)
1 :  for k = 0 → dim(K) − 1 :
2 :    while (∂_k ! = null) do
3 :      while (There is no free face) do
4 :        ∂_k := Remove one cofaces (one column in matrix) in ∂_k and
           label it as critical
5 :      ∂_k := Remove a free face with its coface (Remove ith row and jth
           column)
6 :      ∂_{k−1} := update ∂_{k−1} by removing the ith column
7 :      ∂_{k+1} := update ∂_{k+1} by removing the jth row
8 :      M_k(j) = i
9 :      M_{k−1}(i) = −j
```

Table 3.1 Morse Matching Based on Elementary Collapses

Example 3.1.1. *Consider the Morse matching in figure 2.3, the matching vectors are:*

$$\mathcal{M}_0 = (0, -1, -2, -3),\ \mathcal{M}_1 = (2, 3, -1, 4, 0),\ \mathcal{M}_2 = (3)$$

Example 3.1.2. *Consider the simplicial complex in figure 2.2 and the boundary operator matrices:*

$$\partial_1 = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix},\ \partial_2 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

With respect to the algorithm 3.1, the initial matching vectors are:

$$\mathcal{M}_0 = (0, 0, 0, 0),\ \mathcal{M}_1 = (0, 0, 0, 0, 0),\ \mathcal{M}_2 = (0)$$

As you know each row in ∂_k represents a $(k-1)$-simplex and if in some row, there is just one element equal to 1, that $(k-1)$-simplex is a free face and can be deleted with its coface. Now in ∂_2, there are three free faces, here we match the first 1-simplex with its coface. Therefore δ_2 become a zero dimensional matrix and

∂_1, $\mathcal{N}_1$ and $\mathcal{N}_2$ will be updated:

$$\partial_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix},\ \mathcal{N}_1 = (-1,\ 0,\ 0,\ 0,\ 0),\ \mathcal{N}_2 = (1)$$

Now in ∂_1, the first 0-simplex is a free face, so we match it with its coface and ∂_1, $\mathcal{N}_0$ and $\mathcal{N}_1$ will be updated to:

$$\partial_1 = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix},\ \mathcal{N}_0 = (-2,\ 0,\ 0,\ 0),\ \mathcal{N}_1 = (-1,\ 1,\ 0,\ 0,\ 0)$$

Now as you see, there is no free face. So we have to delete a coface(a column in ∂_1) as critical. Here, we delete the first coface and the updated ∂_1 will be:

$$\partial_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}$$

Now, we have two free faces for matching. We match the first 0-simplex with first 1-simplex and ∂_1, $\mathcal{N}_0$ and $\mathcal{N}_1$ will become:

$$\partial_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix},\ \mathcal{N}_0 = (-2,\ -4,\ 0,\ 0),\ \mathcal{N}_1 = (-1,\ 1,\ 0,\ 2,\ 0)$$

Finally, we have two free faces and one coface, we match the first face with that coface and we have:

$$\mathcal{N}_0 = (-2,\ -4,\ -5,\ 0),\ \mathcal{N}_1 = (-1,\ 1,\ 0,\ 2,\ 3),$$

Here ∂_1 becomes zero dimensional and so the algorithm is completed. Therefore, with respect to matching vectors the discrete Vector field will be:

$$V = \{([0], [02]), ([1], [13]), ([2], [23]), ([01], [012])\}$$

As you see $c_0 = c_1 = 1$ *and* $c_2 = 0$ *and they are equal to Betti numbers. Therefore, the Morse matching is perfect.*

As you saw in the example above, we did the matching with respect to the order of simplices, but in [10], a random version is presented for this algorithm which means that when there is no free face, we randomly select a facet to delete it as critical. As an example, consider the graph depicted in figure 3.3. We want to find a Morse matching. If we select an edge in random, the results can be different for different edges. If we select the central edge, by following the algorithm, the discrete Morse vector will be $C = (2, 3)$, which is not a perfect Morse matching, but if we select any other of the edges, we will obtain a perfect matching with $C = (1, 2)$. So, each simplex that we label as critical in this algorithm, may change the final discrete Morse vector. This case can be either an advantage or a disadvantage. The disadvantage is that if we want to calculate the homology groups and Betti numbers, since the matching is not perfect, we cannot stop and we need to go on by computing Morse complex.

If we focus on 1-dimensional simplicial complexes, by proposition 2.3.1, we know for a graph a perfect Morse Matching always exists. The other method that we will explain in the next section, guarantees the perfect Morse Matching for a graph.

The non perfect Morse Matching obtained by this method can be informative. Let us compare the graphs in figures 3.3 and 3.4. As you see they have isomorphic homology groups and so have equal Betti vectors which is $(1, 2)$, But there is a difference among them. Since at the beginning, no one have a free face, so we need to remove an edge and label it as critical. If all edges have the same probability to be the first candidate, so the probability of not being concluded to a perfect Morse matching for A_7 is equal to $\frac{1}{7}$, while for A_{m+6} it is $\frac{m}{m+6}$. You see that as m increases, the probability of a non perfect Morse matching will increase as well. In the following, we consider a graph G as a substitution of a 1-dimensional simplicial complex.

Corollary 3.1.1. *Each basic cycle of G must contain at least one critical edge.*

This corollary is a result of a Morse Inequalities stated in theorem 2.0.2.

Definition 3.1.5. *Let G be a graph. A subset* $E' \subseteq E$ *of the edge set of G is called a* k***-cut set*** *if* $\beta_0(G) = k + \beta_0(G - E)$*. If* $E' = \{e\}$ *is a single edge, then e is called a* 1***-cut set*** *of G.*

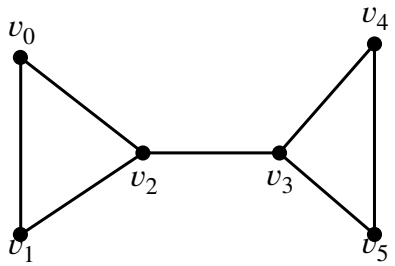


Fig. 3.3 The graph A_7

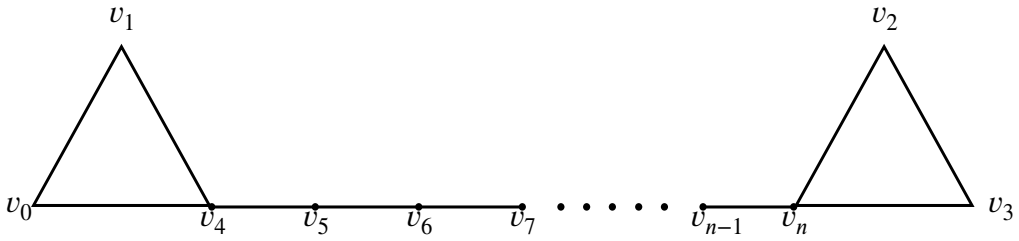


Fig. 3.4 A_{m+6}Similar structure to graph in figure 3.3, but with m edges between the two triangles

In order for E' to be a k-cut set, clearly $|E'| \geq k$.

Proposition 3.1.1. *Let $f : G \to \mathbb{R}$ a discrete Morse function on G, and $k := c_1 - \beta_1 > 0$. Then there exists a subset E'_j of the set of critical edges of f such that E'_j is a j-cut set for every $1 \leq j \leq k$.*

Proof. Let M denote the set of critical edges of f. By Corollary 3.1.1, there is a subset $L \subseteq M$ such that $G - L$ contains no cycles and hence is a forest with $\beta_0(G - L) = \beta_0(G)$ components. Since $|L| = \beta_1(G)$ and $L \subseteq M, |M - L| = k$. Hence the removal of any edge $e \in M - L$ from the graph $G - L$ increases the number of components by 1, and the result follows. □

3.2 Method Based on Elementary Coreduction

We present here another algorithm to find a discrete Morse matching which is called **elementary coreduction** based method. The method was introduced in [16] and

an algorithm based on it was presented in [17]. Along its execution, this algorithm will mark a face as either critical or matched. Once a face has been marked by the algorithm, it is called labeled. Unmarked faces are called unlabeled. This method instead of looking for free faces, looks for simplices that have just one unlabeled 1-codimensional face. If there is no simplex satisfying this condition, the algorithm will label one of the minimal unlabeled faces as critical. One can see an example of this algorithm in figure 3.5. At first, there is no simplex with just one unlabeled face, so one of the minimal faces is labeled as critical, in this case v_1. Then v_2 is matched with $[v_1, v_2]$, and successively, v_3 with $[v_1, v_3]$. Finally, $[v_2, v_3]$ is matched with $[v_1, v_2, v_3]$. Therefore, the only critical simplex is v_1 and the discrete vector field is

$$V = \{([v_2], [v_1, v_2]), ([v_3], [v_1, v_3]), ([v_2, v_3], [v_1, v_2, v_3])\}$$

and the matching vectors will be

$$\mathcal{N}_0 = (0, -1, -2),\ \mathcal{N}_1 = (2, 3, -1),\ \mathcal{N}_2 = (3)$$

The second algorithm is written in table 3.2. In the following, we give two propositions, one that proves algorithm based on coreduction gives a Morse matching, the other one proves that this method guarantees a perfect Morse matching for 1-dimensional simplicial complexes.

Proposition 3.2.1. *The algorithm based on elementary coreduction gives a Morse matching of the modified Hasse diagram of a simplicial complex.*

Proof. Here, we follow the same approach that we used in 3.1. The algorithm always tries to match those simplices that are unlabeled and as soon as they are labeled as either critical or matched, the algorithm do not look at them for matching anymore. So, it is impossible for a simplex to be matched more than one time.

Now, in order to show that this is a Morse matching, we have to show that it is acyclic. We prove this by absurd. Let us assume that we have the following sequence of matchings

$$(\tau_1, \sigma_1), (\tau_2, \sigma_2), \cdots (\tau_n, \sigma_n), \text{ with } \tau_i \lhd \sigma_{i-1} \text{ for } i = 2, \dots, n$$

inducing a path in the modified Hasse diagram. Assume that $\tau_n \lhd \sigma_1$ to obtain a cycle. Observe that, when we matched τ_1 and σ_1, σ_1 has just had one unlabeled

1-codimensional face and this means that its other faces had been earlier marked as either critical or matched. But the matching (τ_n, σ_n) is successive to (τ_1, σ_1), thus the only possibility for τ_n to be a face of both σ_1 and σ_n is that it was critical which is a contradiction.

□

Proposition 3.2.2. *The algorithm based on elementary coreduction guarantees a perfect Morse matching for 1-dimensional simplicial complexes.*

Proof. Let us assume that the simplicial complex K is actually a connected graph G with n vertices and m edges. We will prove that the matched edges are giving us a spanning tree on G. The algorithm for G is exactly working like Prim's algorithm (see [18]), because firstly for this algorithm a vertex is labeled as critical and Prim's algorithm does the same action by choosing an initial vertex, secondly in this algorithm the matching of an unlabeled edge with its single one unlabeled endpoint is exactly the same to the adding an edge into the spanning tree while one of its endpoints is already included in tree and the other is not. Now we know that Prim's algorithm guarantees a spanning tree on G, therefore the Matching algorithm gives an spanning tree, too. So the discrete vector field will be a set of pairs of the $n-1$ edges in the spanning tree and the vertices they are matched with.

Now suppose G is not connected and the components are $G_1, G_2, \cdots, G_n$. The algorithm labels a vertex, say $v \in V(G_1)$, as critical and as proved above, we will obtain a perfect Morse matching on G_1 with a discrete vector field V_1. Afterwards, since the matching is done on G_1 and in other components there is no edge with just one unlabeled face, the algorithm will label a vertex in another component as critical and the it does the matching as it has done for G_1. Therefore, in each component there is just one critical vertex with a discrete Morse vector V_n. The discrete vector field for G will be

$$V = V_1 \cup V_2 \cup \cdots \cup V_n$$

The discrete vector fields for each component have no simplices in common, therefore the union of these discrete vector fields is a disjoint union and V will be a discrete vector field. Moreover, since in each component the matching is acyclic, then in the disjoint union we have no cycle either. □

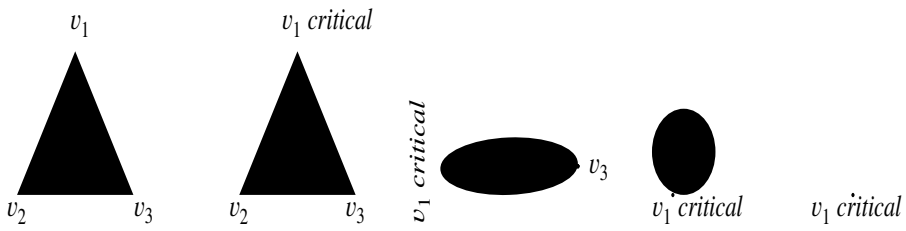


Fig. 3.5 An example of second method of Matching

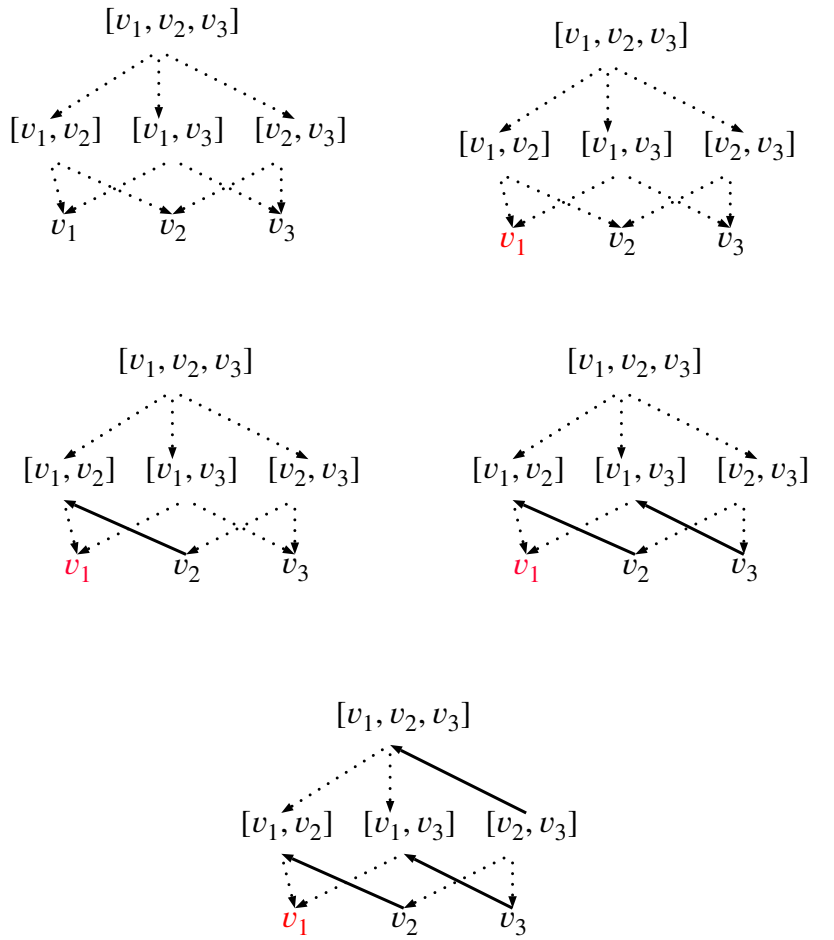


Fig. 3.6 The Process of Matching in figure 3.5 shown on Hasse Diagram

Algorithm 2: Morse Matching Based on Elementary Coreduction

Input: boundary matrices ∂_* of simplicial complex K
Output: matching vectors $\mathcal{N}_*$

0 : Initialize $\mathcal{N}_k = \vec{0}\ \ \forall k = 0, \cdots, dim(K)$
1 : **for** $k = 1 \rightarrow dim(K) - 1$:
2 : **while** $(\partial_k != null)$ **do**
3 : **while** (There is no simplex with just one unlabeled 1-codimensional face) **do**
4 : ∂_k := Remove one face (one row in matrix ∂_k) and label it as critical
5 : ∂_k := Remove a simplex with its unlabeled face(Remove ith row and jth column)
6 : ∂_{k-1} := update ∂_{k-1} by removing the ith column.
7 : ∂_{k+1} := update ∂_{k+1} by removing the jth row.
8 : $\mathcal{N}_k(j) = i$
9 : $\mathcal{N}_{k-1}(i) = -j$

Table 3.2 Morse Matching Based on Elementary Coreduction

3.2.1 Parallel Morse Matching for 2D Simplicial Complex

Here we present a parallel version of the algorithm in table 3.2 for 2-dimensional simplicial complexes which means that at a same time, the Morse matching algorithm is applied to all boundary matrices separately. For this objective, we order the simplices in colexicographical order (see B.0.2) and whenever the algorithm wants to label a simplex either as critical or as matched, it selects the first simplices in colexicographical order.

Since the matching algorithm is applied to all boundary matrices concurrently, it could be possible to have a k-simplex being matched with a $(k-1)$-simplex and a $(k+1)$-simplex at the same time. We call this situation **double matching**. Double matching is not Morse matching and has to be handled in some way. For k-simplices in a simplicial complex K, we have two vectors $\mathcal{N}_k^-$ and $\mathcal{N}_k^+$ whose dimensions are the number of k-simplices in K and the i'th component for each one of them are

Algorithm 3: Parallel Matching for 2-dimensional Simplicial Complex

Input: Colexicographically sorted boundary $m \times n$ matrix ∂_k, $(k = 0, 1)$
Output: Two integer vectors $\mathcal{N}_k$, $\mathcal{N}_{k-1}$ of length n and m respectively

```
0 :  Initialize N_k := (0, ..., 0), N_{k-1} := (0, ..., 0)
1 :  while (∂_k != null) do
2 :    while (There is no column with only one nonzero element) do
3 :      ∂_k := Remove the first row of ∂_k
4 :    Find the first column j with exactly one nonzero element
5 :    ∂_k := Remove the column j with its matched row M(j) = i
6 :    N_k(j) = i
7 :    N_{k-1}(i) = -j
```

Table 3.3 Parallel Matching for 2-dimensional Simplicial Complex

$$\mathcal{N}_k^-(i) = \begin{cases} j & \text{matched with j-th } (k-1)\text{-simplex} \\ 0 & \text{critical} \end{cases} \quad \forall k = 0, \cdots, \dim(K)$$

and

$$\mathcal{N}_k^+(i) = \begin{cases} -j & \text{matched with j-th } (k+1)\text{-simplex} \\ 0 & \text{critical} \end{cases} \quad \forall k = 0, \cdots, \dim(K)$$

If we have double matching for a k-simplex, $\mathcal{N}_k^-(i) \neq 0$ and $\mathcal{N}_k^+(i) \neq 0$, where i is the index of the k-simplex in colexicographical order. In such a situation, we have to do the following steps

1. For each k find the indices i in $\mathcal{N}_k^-$ and $\mathcal{N}_k^+$ where $\mathcal{N}_k^-(i) \neq 0$ and $\mathcal{N}_k^+(i) \neq 0$
2. Set $\mathcal{N}_k^+(i) = 0$ and $\mathcal{N}_{k+1}^-(M(i)) = 0$, where $M(i)$ is the index of the $(k+1)$-simplex matched with ith k-simplex.
3. We set $\mathcal{N}_k = \mathcal{N}_k^- + \mathcal{N}_k^+$

Step (3) is the final step that will give us the k-th matching vector. But there is an exception here. For 1-simplices double matching never happens. We mention this property as a proposition in the following.

Proposition 3.2.3. *For algorithm 3, no double matching happens.*

Proof. At first, we need to explain a little about the figures that we will use for this proposition and also proposition 3.2.4. In these figures $1 \longrightarrow 1'$ means that $1'$ has been removed from matrix (as either for a critical labeling or for a matching) not later than 1. This diagrams help us to make the proofs more intuitive and show how we end up to a contradiction.

In the following we denote $(\sigma, \tau) \in V$ by $\sigma \rightarrow \tau$. What we want to prove is that for 1−simplices double matching never happens. Suppose there is at least one 1−simplex matched to a 0− and a 2−simplex. Let ϕ be double matched and maximal with respect the partial order among the double matched 1-simplices:

$$\phi = [v_i,\ v_j] \quad \text{with } v_i <\ v_j$$

Matching of $[v_i, v_j]$ can have one of the following five situations

$$\begin{array}{ll} Case\ 1 & [v_i] \rightarrow [v_i, v_j] \\ Case\ 2 & [v_j] \rightarrow [v_i, v_j] \\ Case\ A & [v_i, v_j] \rightarrow [v', v_i, v_j] \\ Case\ B & [v_i, v_j] \rightarrow [v_i, v', v_j] \\ Case\ C & [v_i, v_j] \rightarrow [v_i, v_j, v'] \end{array}$$

where v' is a vertex such that, in Case A, $v' < v_i$, in Case B, $v_i < v' < v_j$ and in Case C, $v_j < v'$.

Hence, we have 6 possibilities of having a double matching which are 1.A, 1.B,1.C, 2.A, 2.B, 2.C. Here, we prove one by one that these double matching never happens.

Since the proof is to some extent complicated, for each case, we follow the proof with a graphical presentation to show that how each case ends up to a contradiction.

Case 1.A:

Because we have $[v_i] \rightarrow [v_i, v_j]$ this means that $[v_j]$ has been already deleted by the algorithm. Deleting a face, means that the simplex (in this case the vertex) is already marked. It could be marked either as critical or as matched. Since $[v_j] > [v_i]$, it is thus impossible that the algorithm has marked it as critical, thus $[v_j]$

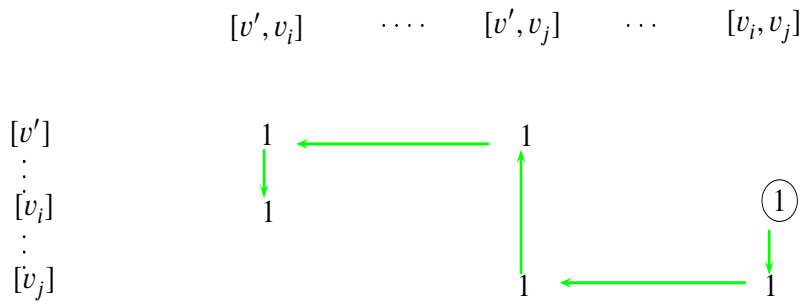


Fig. 3.7 ∂_1 for Case 1.*A*

is matched. Since $[v_j]$ has been marked before, so it has to be matched with one of its 1-codimensional faces σ, where in colexicographical order $\sigma < [v_i, v_j]$. So, for σ we have:

$$\exists v_h \text{ such that } [v_j] \to [v_h, v_j].$$

On the other hand, since $[v', v_i] < [v', v_j] < [v_i, v_j]$ and $[v_j]$ has been marked earlier, so the other face of $[v', v_j]$, $[v']$ should have been marked before $[v_j]$. So $[v']$ should have been marked before $[v_i]$. So with respect to the algorithm, $[v', v_i]$ is the legitimate simplex to be matched with $[v_i]$, that is a contradiction. In figure 3.7, the path that leads us to contradiction is shown.

Case1.B:

In this case we have $v_i < v' < v_j$ and so for 1-simplices $[v_i, v'] < [v_i, v_j] < [v', v_j]$.
$[v_i, v_j] \to [v_i, v', v_j]$ means that the other two faces of $[v_i, v', v_j]$ are already marked. For $[v', v_j]$, since in colexicographical order it comes after $[v_i, v_j]$, it has to be marked as a matched. So $[v', v_j]$ has a coface σ, where $\sigma < [v_i, v', v_j]$. So for σ, we have the following:

$$\exists\, v_h < v_i \,\text{such that}\, [v', v_j] \to [v_h, v', v_j]$$

Now, consider the 1-codimensional faces of $[v_h, v', v_j]$ which are $[v_h, v'] < [v_h, v_j] < [v', v_j]$. So we have:

$$[v_h, v'] < [v_i, v'] < [v_h, v_j] < [v_i, v_j] < [v', v_j]$$

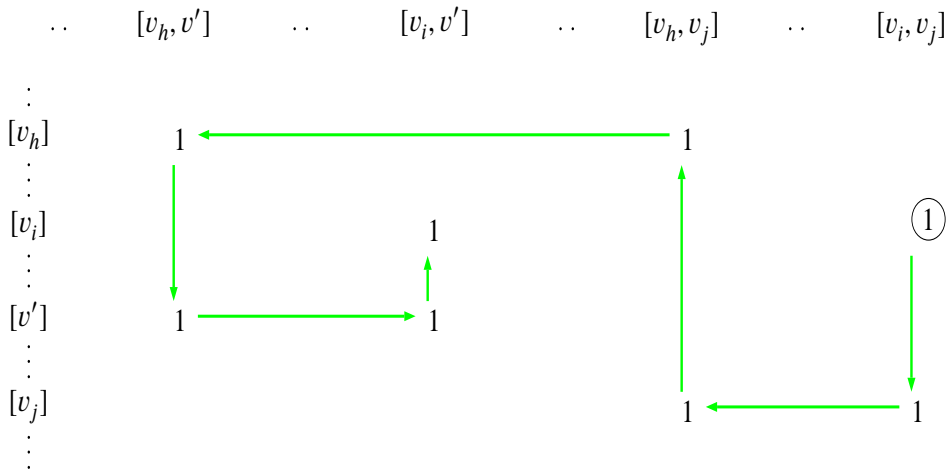


Fig. 3.8 ∂_1 for Case 1.***B***

Now we get back to the matching process of ∂_1. Similar to *Case 1.A*, $[v_j]$ should be already matched with some other 1-simplex. Since $[v_j]$ is one of the faces of $[v_h, v_j]$, so the other 0-simplex $[v_h]$ should be marked before $[v_i]$ either as critical or as matched. Since $[v_h]$ is the face of $[v_h, v']$, so it means that $[v']$ has already at least one coface $[v_h, v']$, that already can be matched with and this matching happens before $[v_i] \rightarrow [v_i, v_j]$. But in this situation $[v_i, v']$ is left with one unmarked face which is $[v_i]$ and by referring to the algorithm, $[v_i, v']$ is the legitimate 1-simplex to be matched with $[v_i]$. So we end up here to another contradiction. In figure 3.8, you see the path that leads to another matching which is a contradiction

Case1.C: It is similar to Case1.B. We have that $v_i < v_j < v'$ and for $1-$simplices we have

$$[v_i, v_j] < [v_i, v'] < [v_j, v'].$$

This means that when we match $[v_i, v_j]$ and $[v_i, v_j, v']$, $[v_i, v']$ and $[v_j, v']$ have been already matched with their cofaces in the following form:

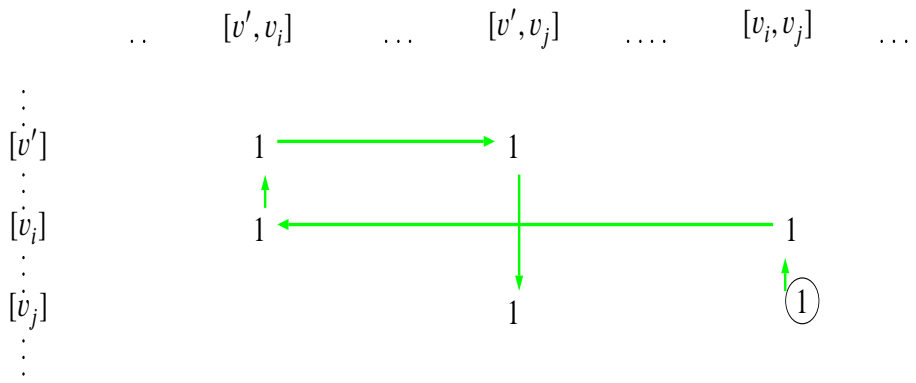


Fig. 3.9 ∂_1 for Case 2.*A*

$$[v_i, v'] \to [v_g, v_i, v'], \quad v_g < v_i$$
$$[v_j, v'] \to [v_h, v_j, v'], \quad v_h < v_j$$

Similar to Case1.B we have that the edge $[v_g, v_i] < [v_i, v_j]$ must have the two unlabeled faces, and it follows the same for the edge $[v_g, v'] < [v_i, v_j]$, the edge $[v_h, v'] < [v_i, v_j]$ and finally the edge $[v_h, v_j] < [v_i, v_j]$. This produces is a contradiction because since we are in Case1, the vertex $[v_j]$ must be matched before $[v_i]$.

Case2.A: In this case we have $[v', v_i] < [v', v_j] < [v_i, v_j]$. Because $[v_j] \to [v_i, v_j]$ the vertex $[v_i]$ was already marked. The fact that $[v', v_i] < [v_i, v_j]$ ensures that the vertex $[v']$ was marked before $[v_j]$. We obtain that $[v', v_j]$ is a better candidate with respect to $[v_i, v_j]$. But this is not possible.

Case2.B:

$$[v_i, v'] < [v_i, v_j] < [v', v_j]$$

Because $[v_i, v_j] < [v', v_j]$ then $\exists v_h$ such that $v_h < v_i$ and $[v', v_j] \to [v_h, v', v_j]$ (the column corresponding to $[v', v_j]$ must be already marked). In particular, because $[v_i, v'] < [v_i, v_j]$ and $[v_i]$ is already marked, we must have $[v']$ already marked as well. Following the same reasoning, we have $[v_h, v'] < [v_i, v_j]$ and $[v']$ marked, that

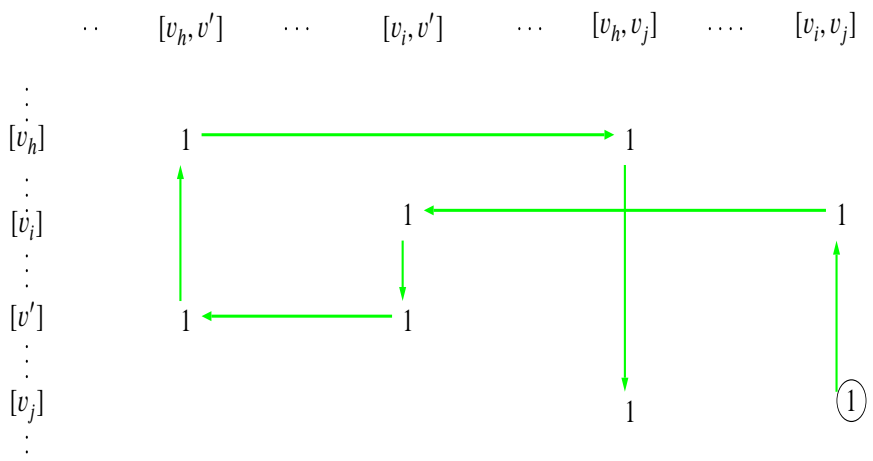


Fig. 3.10 ∂_1 Case 2.*B*

implies $[v_h]$ marked. Finally because $[v_h, v_j] < [v_i, v_j]$ with $[v_h]$ already marked we obtain a contradiction.

Case2.C:

$$[v_i, v_j] < [v_i, v'] < [v_j, v']$$

Similar to Case2.B, because $[v_i, v_j] < [v_i, v']$ and $[v_i, v_j] < [v_j, v']$ it follows that $\exists v_g, v_h$ such that $v_g < v_i,\ v_h < v_i$ and $[v_i, v'] \to [v_g, v_i, v'],\ [v_j, v'] \to [v_h, v_j, v']$. Following the same scheme of Case2.B we obtain the following:

$$\begin{aligned}
&[v_i] \quad \text{marked} \quad [v_g, v_i] < [v_i, v_j] \implies \\
&[v_g] \quad \text{marked} \quad [v_g, v'] < [v_i, v_j] \implies \\
&[v'] \quad \text{marked} \quad [v_h, v'] < [v_i, v_j] \implies \\
&[v_h] \quad \text{marked} \quad [v_h, v_j] < [v_i, v_j] \ \text{contradiction.}
\end{aligned}$$

Therefore, we conclude that it is impossible to have a double matching. □

Example 3.2.1. *Consider an empty tetrahedron. We want to find a Morse matching by using the semi parallel algorithm shown in table 3.3. If* $v_1 < v_2 < v_3 < v_4$, *then colexicographical order* $[v_1, v_2] < [v_1, v_3] < [v_2, v_3] < [v_1, v_4] < [v_2, v_4] <$

$[v_3, v_4]$, *and for 2-simplices* $[v_1, v_2, v_3] < [v_1, v_2, v_4] < [v_1, v_3, v_4] < [v_2, v_3, v_4]$. *So the boundary matrices will be:*

$$\partial_1 = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}, \ \partial_2 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

We apply matching algorithm in table 3.3 separately for ∂_1 *and* ∂_2.

For ∂_1, *at fist, since we do not have any unlabeled 1-simplex with just one unlabeled face, we mark(delete the correspondent row) the first simplex* v_0, *as critical, so we have:*

$$\partial_1 = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}, \ \mathcal{N}_0 = (0, 0, 0, 0), \ \mathcal{N}_1^- = (0, 0, 0, 0, 0, 0)$$

Now the first 1-simplex has just one unlabeled face, so they can get matched and ∂_1, $\mathcal{N}_0$ *and* $\mathcal{N}_1$ *will be updated to:*

$$\partial_1 = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}, \ \mathcal{N}_0 = (0, -1, 0, 0), \ \mathcal{N}_1^- = (2, 0, 0, 0, 0, 0)$$

Again, we can match the first 1-simplex with first 0-simplex in ∂_1. *We have:*

$$\partial_1 = \begin{bmatrix} 0 & 1 & 1 & 1 \end{bmatrix}, \ \mathcal{N}_0 = (0, -1, -2, 0), \ \mathcal{N}_1^- = (2, 3, 0, 0, 0, 0)$$

and finally, we match the second 1-simplex with the remaining 0-simplex:

$$\mathcal{N}_0 = (0, -1, -2, -4), \ \mathcal{N}_1^- = (2, 3, 0, 4, 0, 0)$$

Now we do the matching for ∂_2. For having at least one 2-simplex with just one unlabeled face, we need to label the first two 1-simplices as crtical. So ∂_2 will be:

$$\partial_2 = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}, \ \mathscr{N}_1^+ = (0,0,0,0,0,0), \ \mathscr{N}_2^- = (0,0,0,0)$$

Now we can match the first 2-simplex with the first 1-simplex and we will have:

$$\partial_2 = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}, \ \mathscr{N}_1^+ = (0,0,-1,0,0,0), \ \mathscr{N}_2^- = (3,0,0,0)$$

Now there is no 2-simplex with just one unlabeled 1-simplex. So we mark the first 1-simplex as critical and ∂_2 will become:

$$\partial_2 = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

Then we match the first 2-simplex with the first 1-simplex:

$$\partial_2 = \begin{bmatrix} 1 & 1 \end{bmatrix}, \ \mathscr{N}_1^+ = (0,0,-1,0,-2,0), \ \mathscr{N}_2^- = (3,5,0,0)$$

Finally, we match the first 2-simplex with remaining 1-simplex:

$$\mathscr{N}_1^+ = (0,0,-1,0,-2,-3), \ \mathscr{N}_2^- = (3,5,6,0)$$

By looking at $\mathscr{N}_1^-$ and $\mathscr{N}_1^+$, you see no double matching has happened, because there is no 1-simplex τ such that $\mathscr{N}_1^-(\tau) \neq 0$ and $\mathscr{N}_1^+(\tau) \neq 0$. Finally the matching vectors will be:

$$\begin{gathered} \mathscr{N}_0 = (0,-1,-2,-4), \\ \mathscr{N}_1 = \mathscr{N}_1^- + \mathscr{N}_1^+ = (2,3,-1,4,-2,-3), \\ \mathscr{N}_2 = \mathscr{N}_2^- = (3,5,6,0) \end{gathered}$$

Algorithm 4: Parallel Morse Matching

Input: Colexicographically Sorted boundary $m \times n$ matrix ∂_k
Output: Two integer vectors $\mathcal{N}_k$, $\mathcal{N}_{k-1}$ of length n and m respectively

```
0 :  Initialize N_k := (0, ..., 0), N_{k-1} := (0, ..., 0)
1 :  ∂_k := In each column change the first k − 1 1's to 0 such that we have
     just two 1's in each column
2 :  while (∂_k != null) do
3 :    while (There is no column with only one nonzero element) do
4 :      ∂_k := Remove the first row of ∂_k
5 :    Find the first column j with exactly one nonzero element
6 :    ∂_k := Remove the column j with its matched row M(j) = i
7 :    N_k(j) = i
8 :    N_{k-1}(i) = −j
```

Table 3.4 Parallel Morse Matching

3.2.2 Parallel Morse Matching Algorithm

The algorithm described in table 3.3, is parallel for 2-dimensional simplicial complexes, but for higher dimensions we need to check if there is any double matching. Now, we introduce a new version of the algorithm that can be parallelized for all dimensions.

The parallel algorithm is written in table 3.4. What we generally do here is that each simplex can be matched just with its two last 1-codimensional faces. Consider that the simplices are ordered in colexicographical order.

Proposition 3.2.4. *In algorithm 4 written in table 3.4, no double matching happens.*

Proof. Consider a k-simplex $\tau = [v_{i_0}, v_{i_1}, \cdots, v_{i_k}]$. We want to prove that it is impossible for τ to be matched at the same time with a $(k-1)$ and a $(k+1)$-simplex. With respect to the algorithm τ can be matched with just one of two faces:

$$\phi_1 = [v_{i_0}, v_{i_2}, v_{i_3} \cdots, v_{i_k}] \; \phi_2 = [v_{i_1}, v_{i_2}, v_{i_3} \cdots, v_{i_k}]$$

On the other hand, it may have proper cofaces with two following form:

$$\sigma_1 = [v_j, v_{i_0}, v_{i_1}, \cdots, v_{i_k}] \; \forall j < i_0$$
$$\sigma_2 = [v_{i_0}, v_j, v_{i_1}, \cdots, v_{i_k}] \; \forall i_0 < j < i_1$$

Because of the procedure of the algorithm, it cannot be matched with other cofaces. Four situations can happen that make the algorithm fail. These four situations are four forms of double matching. Here, we prove one by one that none of them happen in this algorithm.

Note: $\phi \to \sigma$ notation here means that ϕ is matched with σ and $\phi \lhd \sigma$.

1. $\phi_1 \to \tau \to \sigma_1$

   $\phi_1 \to \tau$ means that ϕ_2 had been marked before as matched. ϕ_2 cannot be critical, because with respect to the algorithm, a simplex will be marked as a critical when there is no coface with exactly one unlabeled face. If we had such a situation before, regarding to the algorithm, ϕ_1 has the priority compared to ϕ_2. So, if ϕ_2 is already marked, it is matched to another k-simplex:

   $$\phi_2 \to \tau' = [v', v_{i_1}, v_{i_2} \cdots, v_{i_k}](v' < v_{i_0})$$

   In the algorithm, in the step that τ is matched with ϕ_1, τ has to be the first unlabeled k-simplex with only one unlabeled face. So the k-simplex $[v_j, v_{i_1}, v_{i_2} \cdots, v_{i_k}]$ is already marked and therefore it implies $[v_j, v_{i_2}, v_{i_3} \cdots, v_{i_k}]$ is already marked as well. But it is impossible, because in this case, the k-simplex $[v_j, v_{i_1}, v_{i_2} \cdots, v_{i_k}]$ could be matched with $\phi_1 = [v_{i_0}, v_{i_2}, v_{i_3} \cdots, v_{i_k}]$ before τ. That's a contradiction.

   Figure 3.11 shows the procedure of proof and the contradiction that finally we end to. Every arrow in the picture shows that the element which the arrow is pointing to, had to be removed(as either critical or matched) earlier than the element that arrow is coming from.

2. $\phi_1 \to \tau \to \sigma_2$

$$[v_j, v_{i_0}, v_{i_2}, v_{i_3}, \cdots v_{i_k}] \cdots [v_j, v_{i_1}, v_{i_2}, \cdots v_{i_k}] \cdots [v_{i_0}, v_{i_1}, v_{i_2} \cdots, v_{i_k}]$$

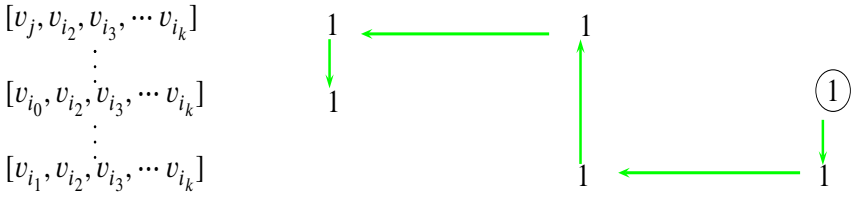


Fig. 3.11 ∂_k for First Case

$\tau \rightarrow \sigma_2$ implies that the other face of σ_2 which is $[v_j, v_{i_1}, v_{i_2}, \cdots, v_{i_k}] > \tau$ has been matched earlier. Therefore, there is a $(k+1)$-simplex in the form of $[v', v_j, v_{i_1}, v_{i_2}, \cdots, v_{i_k}] < \sigma_2$, (such that $v' < v_{i_0}$) which is matched to $[v_j, v_{i_1}, v_{i_2}, \cdots, v_{i_k}]$.

Right now let's consider the two particular faces of $[v', v_j, v_{i_1}, v_{i_2}, \cdots, v_{i_k}]$ in ∂_{k-1}; these two are $[v', v_j, v_{i_2}, v_{i_3}, \cdots, v_{i_k}]$ and $[v', v_{i_1}, v_{i_2}, \cdots, v_{i_k}]$. Right now, consider the following ordering:

$$[v', v_j, v_{i_2}, v_{i_3}, \cdots, v_{i_k}] < [v_{i_0}, v_j, v_{i_2}, v_{i_3}, \cdots, v_{i_k}] < [v', v_{i_1}, v_{i_2}, \cdots, v_{i_k}] < [v_{i_0}, v_{i_1}, v_{i_2}, \cdots, v_{i_k}]$$

because $\phi_1 \rightarrow \tau$, so ϕ_2 has to be matched earlier. ϕ_2 is the second face of $[v', v_{i_1}, v_{i_2}, \cdots, v_{i_k}]$ and so its first face which is $[v', v_{i_2}, v_{i_3}, \cdots, v_{i_k}]$ is already marked. Right now, because $[v', v_{i_2}, v_{i_3}, \cdots, v_{i_k}]$ is the first face of $[v', v_j, v_{i_2}, v_{i_3}, \cdots, v_{i_k}]$, so $[v', v_j, v_{i_2}, v_{i_3}, \cdots, v_{i_k}]$ has to be matched to its second face which is $[v_j, v_{i_2}, v_{i_3}, \cdots, v_{i_k}]$. In another hand, $[v_j, v_{i_2}, v_{i_3}, \cdots, v_{i_k}]$ is the second face of $[v_{i_0}, v_j, v_{i_2}, v_{i_3}, \cdots, v_{i_k}]$ and because $[v_j, v_{i_2}, v_{i_3}, \cdots, v_{i_k}]$ is already marked, so ϕ_1 which is the first face of $[v_{i_0}, v_j, v_{i_2}, v_{i_3}, \cdots, v_{i_k}]$ has to be matched to it. Consider that $[v_{i_0}, v_j, v_{i_2}, v_{i_3}, \cdots, v_{i_k}] < \tau$. That's a contradiction.

3. $\phi_2 \rightarrow \tau \rightarrow \sigma_1$

$$.[v', v_j, v_{i_2}, v_{i_3}, \cdots, v_{i_k}] \,.\, [v_{i_0}, v_j, v_{i_2}, v_{i_3}, \cdots, v_{i_k}] \,.\, [v', v_{i_1}, v_{i_2}, \cdots, v_{i_k}] \,.\, [v_{i_0}, v_{i_1}, v_{i_2}, \cdots, v_{i_k}]$$

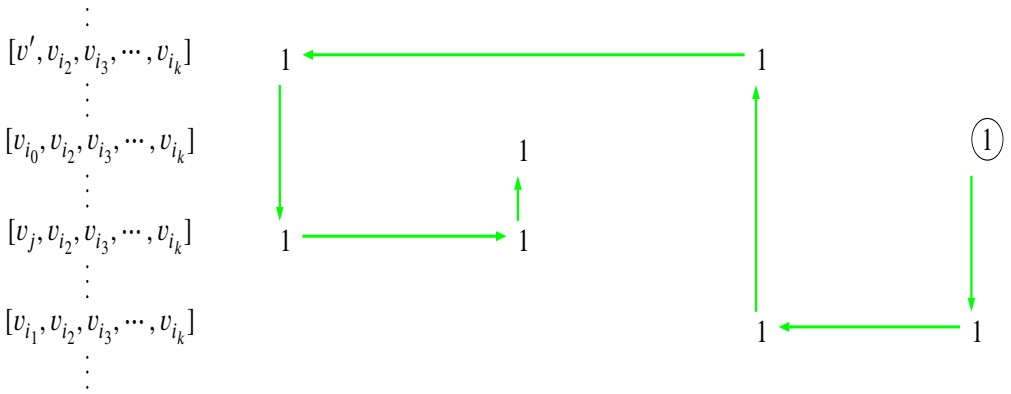


Fig. 3.12 ∂_k for Second Case

$$.[v_j, v_{i_0}, v_{i_2}, v_{i_3}, \cdots, v_{i_k}] \ldots [v_j, v_{i_1}, v_{i_2}, \cdots, v_{i_k}] \ldots . [v_{i_0}, v_{i_1}, v_{i_2}, \cdots, v_{i_k}] \ldots$$

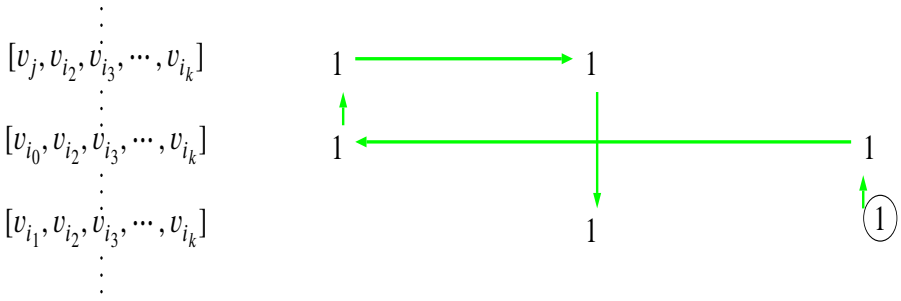


Fig. 3.13 ∂_k for Third Case

By considering the colexicographical order:

$$[v_j, v_{i_0}, v_{i_2}, v_{i_3} \cdots, v_{i_k}] < [v_j, v_{i_1}, v_{i_2}, \cdots, v_{i_k}] < [v_{i_0}, v_{i_1}, \cdots, v_{i_k}]$$

$\phi_2 \to \tau$ means that ϕ_1 has been marked before. You can see, ϕ_1 is the second face of $[v_j, v_{i_0}, v_{i_2}, v_{i_3} \cdots, v_{i_k}]$, So the other face $[v_j, v_{i_0}, v_{i_2}, v_{i_3} \cdots, v_{i_k}]$ which is $[v_j, v_{i_2}, v_{i_3} \cdots, v_{i_k}]$ had to be marked in advance. Besides, $[v_j, v_{i_2}, v_{i_3} \cdots, v_{i_k}]$ is the first face of $[v_j, v_{i_1}, v_{i_2} \cdots, v_{i_k}]$. That's where we get the contradiction, because here the first column which has just one entry equal to 1 is $[v_j, v_{i_1}, v_{i_2} \cdots, v_{i_k}]$, not τ.

4. $\phi_2 \to \tau \to \sigma_2$

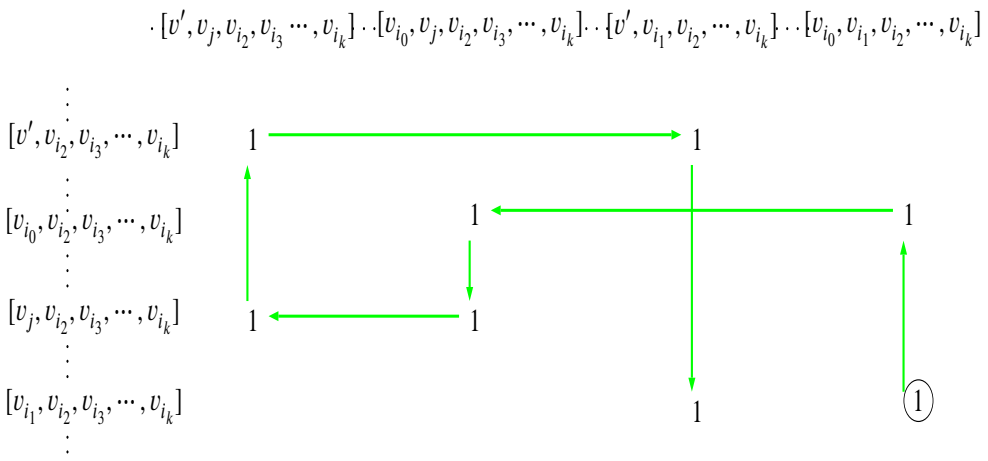


Fig. 3.14 ∂_k for Fourth Case

$\tau \rightarrow \sigma_2$ means that the second face of $\sigma_2, [v_j, v_{i_1}, v_{i_2}, \cdots, v_{i_k}]$ has been matched before. So it should exist another $(k+1)$-simplex before σ_2 such that it is matched to $[v_j, v_{i_1}, v_{i_2}, \cdots, v_{i_k}]$ named $\sigma' = [v', v_j, v_{i_1}, v_{i_2}, \cdots, v_{i_k}]$. But σ' has another face $[v', v_{i_1}, v_{i_2}, \cdots, v_{i_k}] < [v_j, v_{i_1}, v_{i_2}, \cdots, v_{i_k}]$ that had to be marked earlier and so as you may see in this case σ' has to be matched. Right now, consider tow faces of σ' which are $[v', v_j, vi_2, v_{i_3} \cdots, v_{i_k}]$ and $[v', v_{i_1}, v_{i_2}, \cdots v_{i_k}]$. Consider the following colexicographical order in ∂_{k-1}:

$$[v', v_j, v_{i_2}, v_{i_3}, \cdots, v_{i_k}] < [v_{i_0}, v_j, v_{i_2}, v_{i_3}, \cdots, v_{i_k}] < [v', v_{i_1}, v_{i_2}, \cdots, v_{i_k}] < [v_{i_0}, v_{i_1}, v_{i_2}, \cdots, v_{i_k}]$$

$\phi_2 \rightarrow \tau$ means that ϕ_1 is already marked. ϕ_1 is the first face of $[v_{i_0}, v_j, v_{i_2}, v_{i_3}, \cdots, v_{i_k}]$, so here $[v_{i_0}, v_j, v_{i_2}, v_{i_3}, \cdots, v_{i_k}]$ can be matched to its second face $[v_j, v_{i_2}, v_{i_3}, \cdots, v_{i_k}]$. In addition, $[v_j, v_{i_2}, v_{i_3}, \cdots, v_{i_k}]$ is the second face of $[v', v_j, v_{i_2}, v_{i_3}, \cdots, v_{i_k}]$ and its first face $[v', v_{i_2}, v_{i_3}, \cdots, v_{i_k}]$ has to be already marked. In another hand, $[v', v_{i_2}, v_{i_3}, \cdots, v_{i_k}]$ is the first face of the $[v', v_{i_1}, v_{i_2}, v_{i_3}, \cdots, v_{i_k}]$ and the second face is ϕ_2. As you see this $(k+1)$-simplex is the better candidate (look at colexicographical order) to be matched to ϕ_2. That's the contradiction.

□

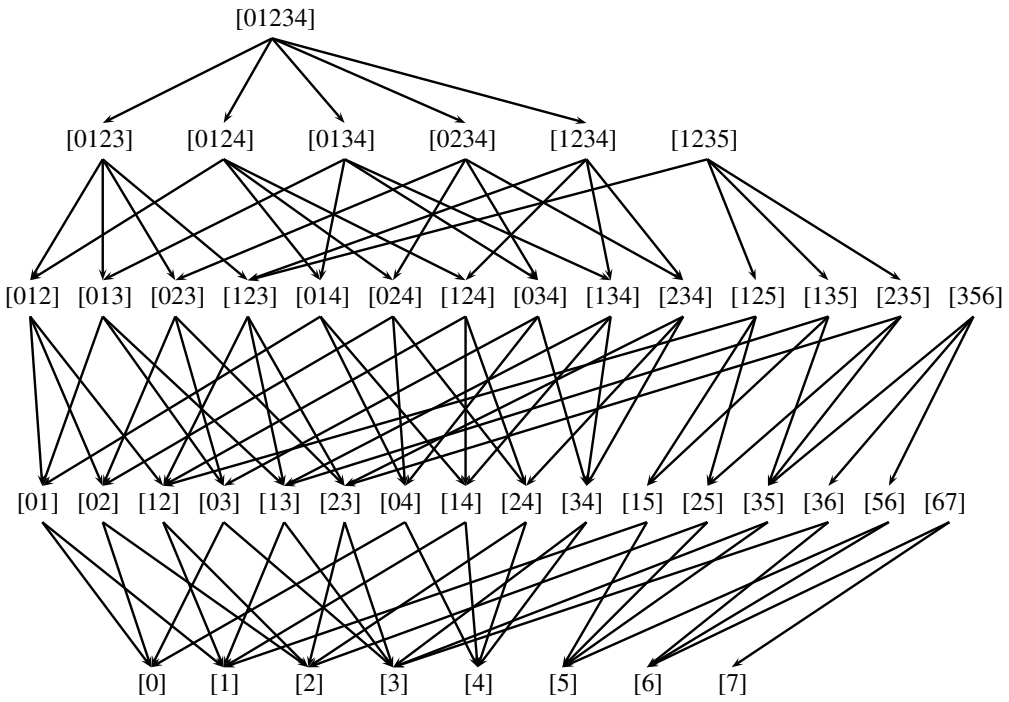


Fig. 3.15 Hasse Diagram of a Simplicial Complex

3.2.3 An Example Of Parallel Morse Matching

In this subsection, we give a big example for parallel Morse matching. We show the whole procedure on Hasse diagram. In figure 3.15, there is the Hasse Diagram of a 4 dimensional simplicial complex. Then in figures 3.16 and 3.17, we remove the first $n-1$ faces for any n-simplex. Then we split up the Hasse Diagram into different layers and we do the matching for each layer separately. This procedure is presented in figures 3.18, 3.19, 3.20 and 3.21. Then, as shown in figure 3.22, after the matching is done, all layers will get back together and we have the complete schema of matching. In all modified Hasse diagrams, those simplices with underlines are critical simplices.

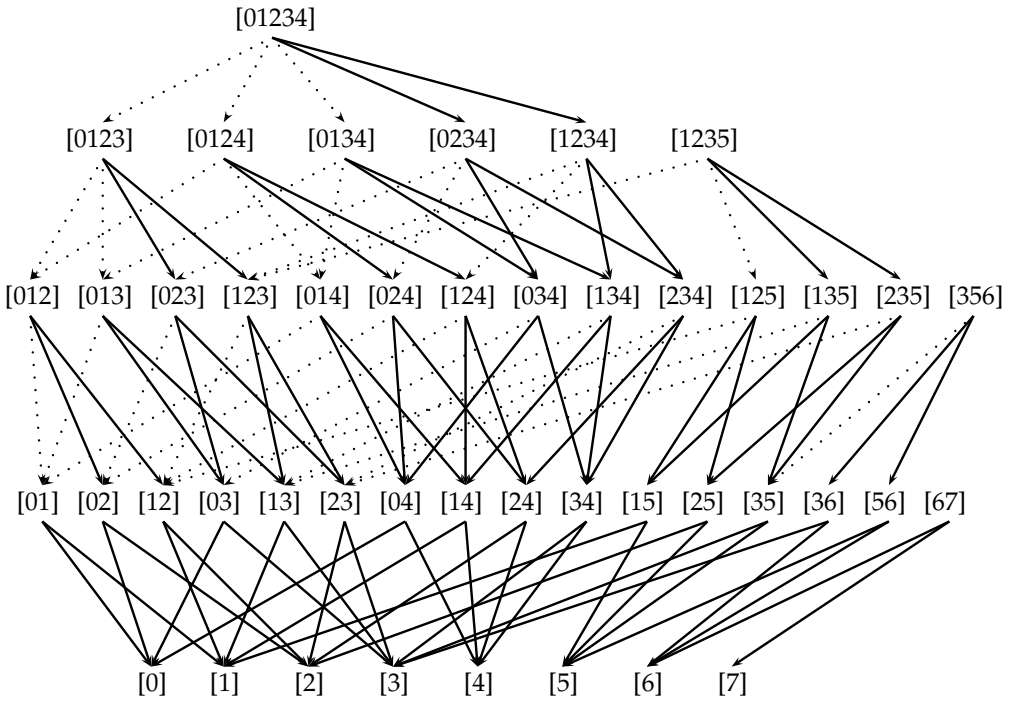


Fig. 3.16 Hasse Diagaram of the Simplicial Complex in Fig.3.15. Here the dashed lines are those that have to be removed with respect to the algorithm in Table.3.4

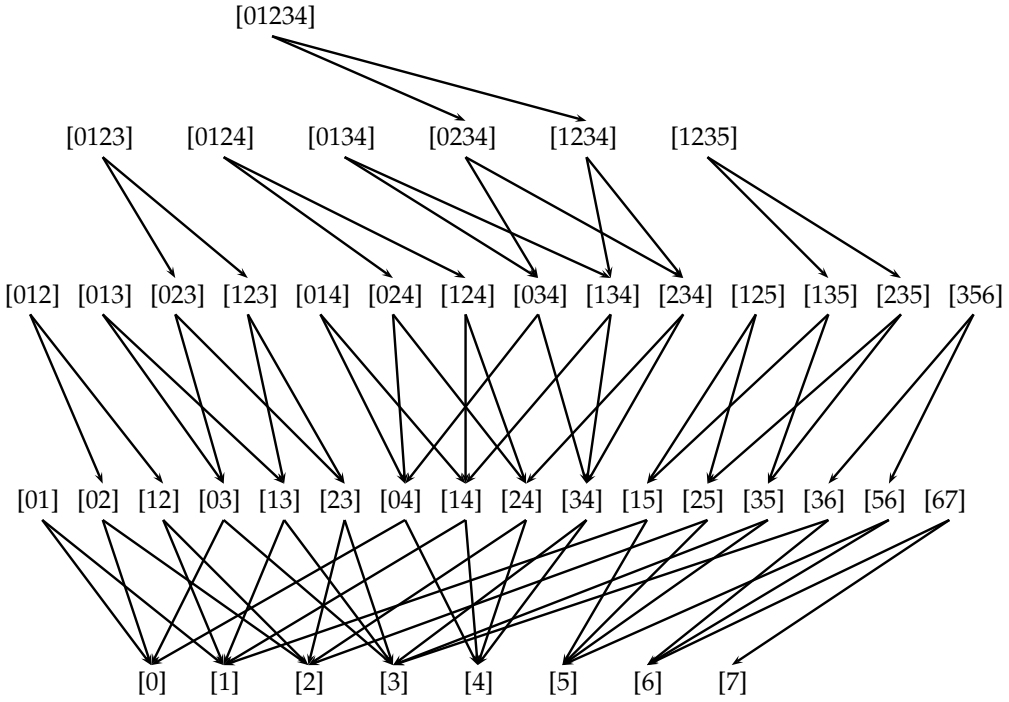


Fig. 3.17 The Hasse Diagram with removed arcs, this stucture can be parallelized

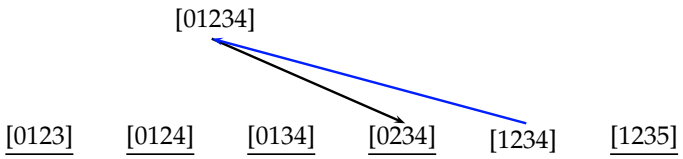


Fig. 3.18 Morse Matching for ∂_4

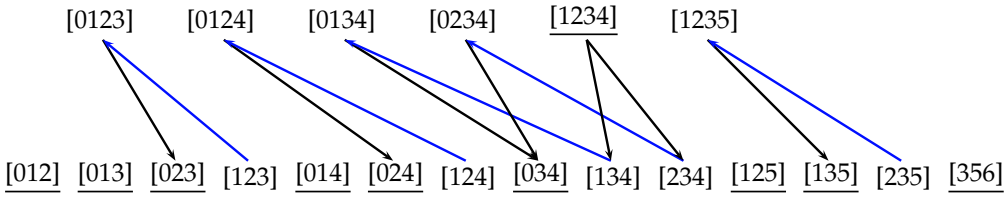


Fig. 3.19 Morse Matching for ∂_3

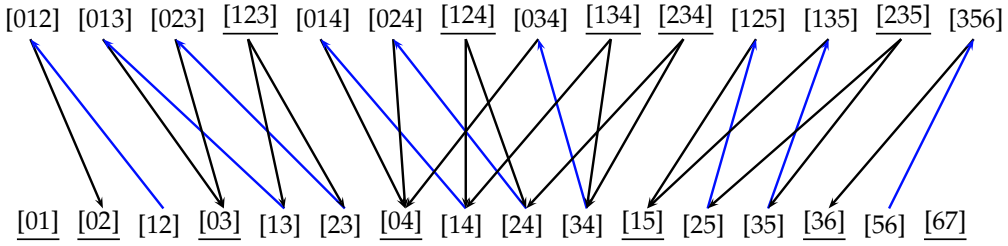


Fig. 3.20 Morse Matching for ∂_2

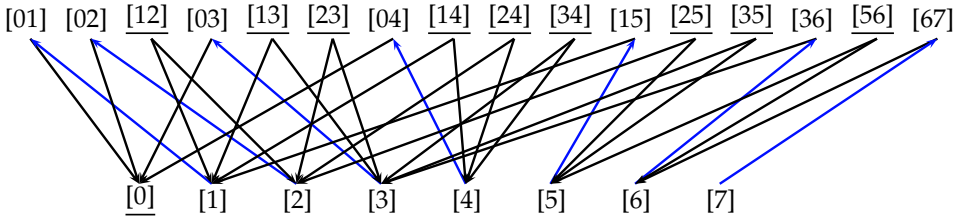


Fig. 3.21 Morse Matching for ∂_1

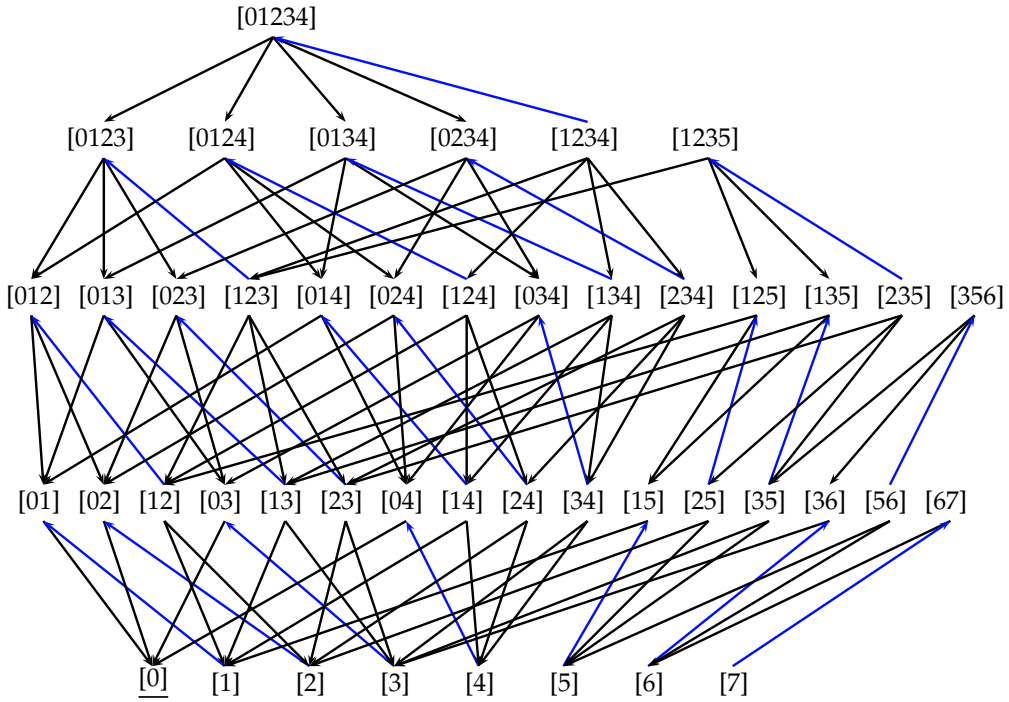


Fig. 3.22 Final result of Matching with just one critical cell [0]

Chapter 4

Application of Discrete Morse Theory in Homology

In chapter 3, we showed two methods for Morse matching computation. In this chapter, we will explain how to calculate the homology groups and Betti numbers based on Morse matching. Using discrete Morse theory for computing homology groups was given for the first time by Lewiner in [19].

In last chapter, we showed that for a 1-dimensional simplicial complex, there is always a perfect Morse matching and in section 3.2, we described an algorithm that always gives a perfect Morse matching. But if the Morse matching is not perfect, we need to calculate the Morse complex by calculating the boundary operator matrices $\partial_k^{\mathscr{M}}$. In section 2.2, we explained how to compute Morse boundary operators.

In the first section, we will briefly explain two methods for Morse boundary computation, first method is based on constructing an adjacency matrix and the second method is based on the modified Hasse diagram.

In the second section, we present a new method that merges computation of Morse matching and Morse boundary operator matrices which means that as long as the matching algorithm is being executed, the Morse boundary matrices are computed at the same time.

4.1 Morse Boundary

After calculating the matching vectors $\mathcal{N}_k\,(k = 0, 1, \cdots, dim(K))$, we have to construct the Morse boundary operator matrices $\partial_k^{\mathcal{M}}$ through V-paths computation.

Consider a boundary matrix ∂_k which is an $m \times n$ matrix with two matching vectors $\mathcal{N}_{k-1}$ and $\mathcal{N}_k$ with lengths m and n, respectively. The number of zeros in $\mathcal{N}_{k-1}$ and $\mathcal{N}_k$ indicate how many $(k-1)$- and k-critical simplices exist, respectively. Suppose there are m_c and n_c critical simplices, so $\partial_k^{\mathcal{M}}$ will be an $m_c \times n_c$ matrix. Since we are working on $\mathbb{Z}_2$ coefficients, $\partial_k^{\mathcal{M}}(i, j)$ is equal to 1, if the number of V-paths between the i-th $(k-1)$-critical simplex and j-th k-critical simplex is an odd number, otherwise that would be 0.

Adjacency Matrix Based Method

In this method, considering the subgraph of modified Hasse diagram that corresponds to dimensions k and $k-1$ of the simplicial complex as a directed graph with $m+n$ vertices, at first we construct the adjacency matrix $\mathcal{A}_k$. The dimension of this matrix is $(m+n) \times (m+n)$. Then we can count the number of paths among any two nodes thanks to the following theorem.

Theorem 4.1.1. *If $\mathcal{A}$ is the adjacency matrix of a graph or digraph $\mathcal{G}$ with vertices $v_1, v_2 \cdots, v_n$, then the i, j entry of $\mathcal{A}^k$ is the number of paths of length k from v_i to v_j*

Proof. See [20], pag. 136,137. □

By this algorithm, the number of paths from node i to j is equal to $\Sigma_t \mathcal{A}^t(i, j)$. In this case, because the number of paths can be just *odd*. So here we can just consider odd exponents of $\mathcal{A}$ and make this algorithm faster.

Method Based on Modified Hasse diagram

In this method, we consider the modified Hasse diagram where the subgraph of vertices that correspond to $(k-1)$- and k-simplices construct a bipartite graph. We divide the algorithm in two sub algorithms. At first, by referring to matching vectors, we do some modifications on boundary matrix, then from that modified matrix, the

next part of algorithm counts the number of paths between critical simplices. The first part of this algorithm is shown in table 4.1.

In the second step, we have the modified boundary matrix as the input and by the algorithm written in table 4.2, the number of V-paths between any pair of critical $(k-1)$- and k-simplices will be found.

Example 4.1.1. *In example 2.2.1, the boundary matrices are*

$$\partial_1 = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}, \quad \partial_2 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

and the matching vectors are

$$\mathscr{N}_0 = (0, 1, 2, 4), \ \mathscr{N}_1 = (-2, -3, 1, -4, 0), \ \mathscr{N}_2 = (-3)$$

By applying the algorithm in table 4.1 to ∂_1, $\mathscr{N}_0$ and $\mathscr{N}_1$, ∂_1 will be modified to

$$\partial_1 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & -1 & 1 \end{bmatrix}$$

and then by using algorithm in table 4.2, the Morse boundary operator matrix will be

$$\partial_1^{\mathscr{M}} = [0]$$

∂_2 after the first algorithm will become zero dimensional.

Algorithm 5a: Boundary Matrix Adjustment

Input: Sorted boundary $m \times n$ matrix ∂_k and matching vectors $\mathcal{N}_{k-1}$ and $\mathcal{N}_k$
Output: Modified boundary matrix, m_c and n_c

```
0 :   Initialize m_c = 0 and n_c = 0
1 :   for i := 1 : m do
2 :     if N_{k-1}(i) = 0
3 :       m_c := m_c + 1
4 :     if N_{k-1}(i) > 0
5 :       ∂_k(i, N_{k-1}(i)) := −1
6 :     if N_{k-1}(i) < 0
7 :       ∂_k :=Remove i-th row of ∂_k
8 :   end
9 :   for i := 1 : n do
10 :    if N_k(i) = 0
11 :      n_c := n_c + 1
12 :    if N_k(i) > 0
13 :      ∂_k :=Remove i-th column of ∂_k
14 :  end
```

Table 4.1 Boundary Matrix Adjustment

Algorithm 5b: Morse Boundary Matrix

Input: Modified Boundary Matrix from algorithm 4.1
Output: Morse boundary matrix $\partial_k^{\mathscr{M}}$

```
0 :   for i := 1 : m_c and j := 1 : n_c do
1 :     if ∂_k(i, j) = 1
2 :       d(i, j) := d(i, j) + 1
3 :     d(i, j) := Σ_l d(l, j)
4 :     ∂_k^M(i, j) := d(i, j)mod2
5 :   end
```

Table 4.2 Morse Boundary Matrix

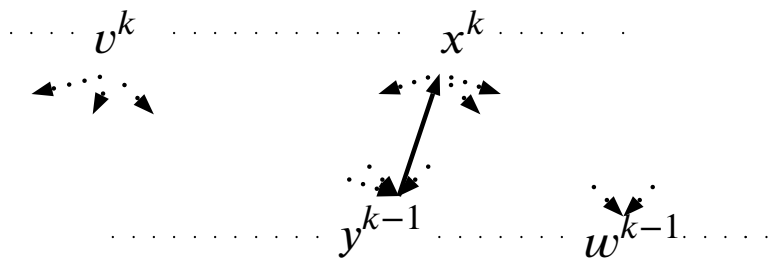


Fig. 4.1 snapshot of when x^k and y^{k-1} are matched and the number of paths among non-matched simplices has to be updated

4.2 Merging Morse Matching and Morse Boundary Computation

In this section, we will merge the two steps of parallel matching and Morse boundary matrix calculation to make a faster algorithm for computation of homology groups and Betti numbers.

Suppose v^k and w^{k-1} are two either critical or unlabeled k- and $(k-1)$-simplices. Let $\nu_t(v^k, w^{k-1})$ show the number of V-paths between these two simplices at step t of the execution of the algorithm 3.4. So in the beginning $\nu_0(v^k, w^{k-1})$ is equal to 1, if w^{k-1} is one of the faces of v^k, otherwise it will be 0. Referring to figure 4.1, when x^k and y^{k-1} are matched in step t, the number of paths in step t will be:

$$\nu_t(v^k, w^{k-1}) = \nu_{t-1}(v^k, w^{k-1}) + \nu_{t-1}(v^k, y^{k-1}) \times \nu_{t-1}(x^k, w^{k-1})$$

When the parallel matching algorithm which is written in table 3.4 (or in table 3.3 for 2-dimensional simplicial complexes) is running, by any matching the boundary matrix will be updated by the above formula. There is a small difference for matching, here whenever in ∂_k, a $(k-1)$-simplex is marked as critical, we do not delete it from ∂_k, because we need critical simplices for Morse boundary matrices, but like before for matching algorithm they are not considered anymore. In the following, by an example, we will explain how the algorithm works.

Example 4.2.1. *In figure 4.3, you see a 2-simplicial complex with its two boundary matrices ∂_1 and ∂_2. Then in figures 4.4, 4.5, you can see step by step how the matching and updating for ∂_1 operate at the same time. You see also the same procedure*

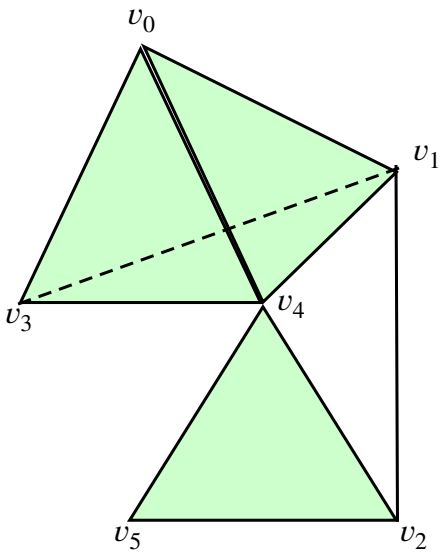


Fig. 4.2 Simplicial Complex $\mathscr{K}$

for ∂_2 in figures 4.6, 4.7 and 4.8. After the matching is finished, the 1-simplices that are matched from the other side will be eliminated and finally we will end up with two Morse boundary matrices $\partial_1^{\mathscr{M}}$ and $\partial_2^{\mathscr{M}}$ represented in figure 4.9.

Here we need to give some explanation about the figures. The arrows indicate that the matching algorithm just look at the boundary matrix from that row to the final row. The rows before that arrow are not considered in matching algorithm. Just circled entries participate in matching algorithm, but for Morse boundary calculation all entries will participate.

∂_1:

	$[v_0,v_1]$	$[v_1,v_2]$	$[v_0,v_3]$	$[v_1,v_3]$	$[v_0,v_4]$	$[v_1,v_4]$	$[v_2,v_4]$	$[v_3,v_4]$	$[v_2,v_5]$	$[v_4,v_5]$
v_0	1		1		1					
v_1	1	1		1		1				
v_2		1					1		1	
v_3			1	1				1		
v_4					1	1	1	1		1
v_5									1	1

∂_2:

	$[v_0,v_1,v_3]$	$[v_0,v_1,v_4]$	$[v_0,v_3,v_4]$	$[v_1,v_3,v_4]$	$[v_2,v_4,v_5]$
$[v_0,v_1]$	1	1			
$[v_1,v_2]$					
$[v_0,v_3]$	1		1	1	
$[v_1,v_3]$	1				
$[v_0,v_4]$		1	1		
$[v_1,v_4]$		1		1	
$[v_2,v_4]$					1
$[v_3,v_4]$			1	1	
$[v_2,v_5]$					1
$[v_4,v_5]$					1

Fig. 4.3 two boundary matrices ∂_1 and ∂_2 of simplicial complex of Fig. 4.2

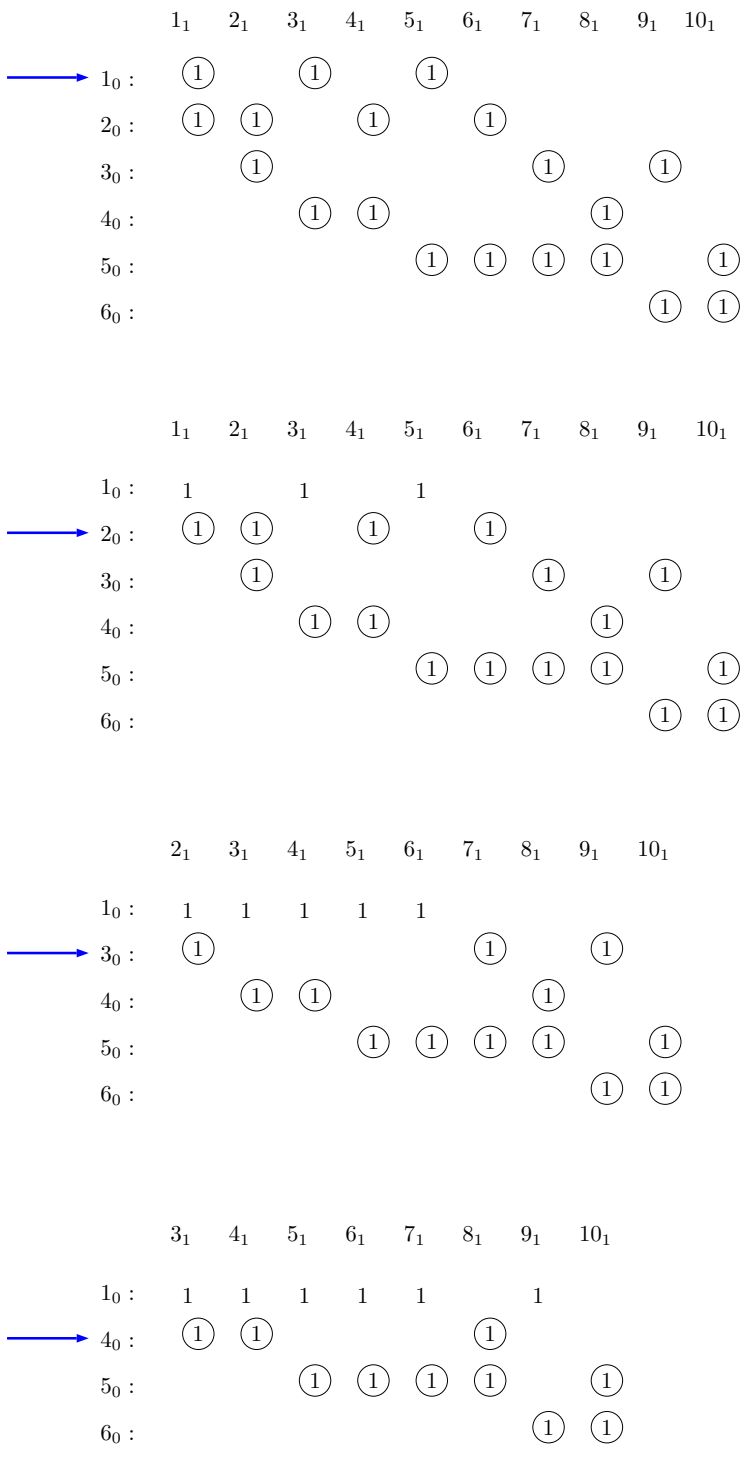


Fig. 4.4 Matching and V-path counting Procedure for ∂_1

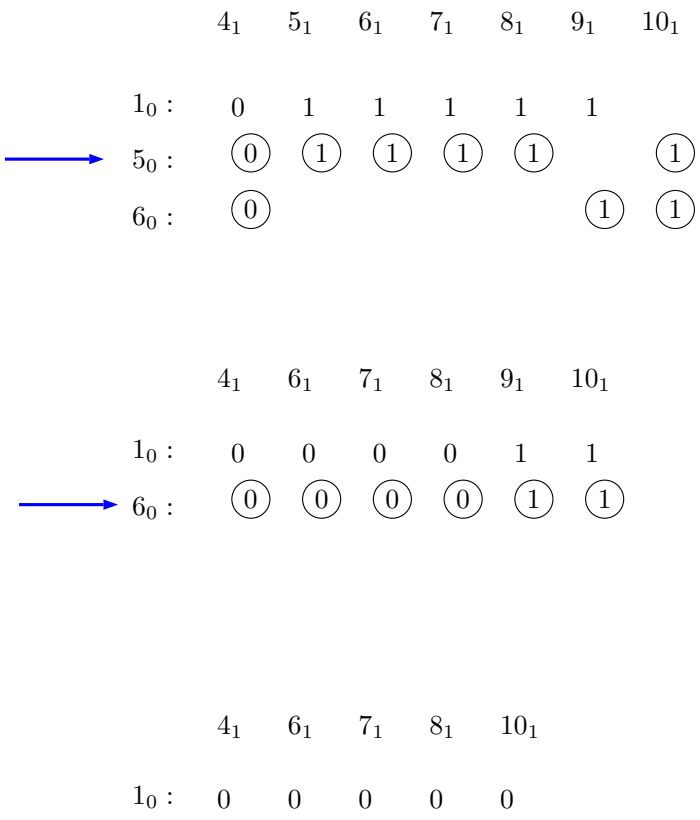


Fig. 4.5 Matching and V-path Procedure for ∂_1

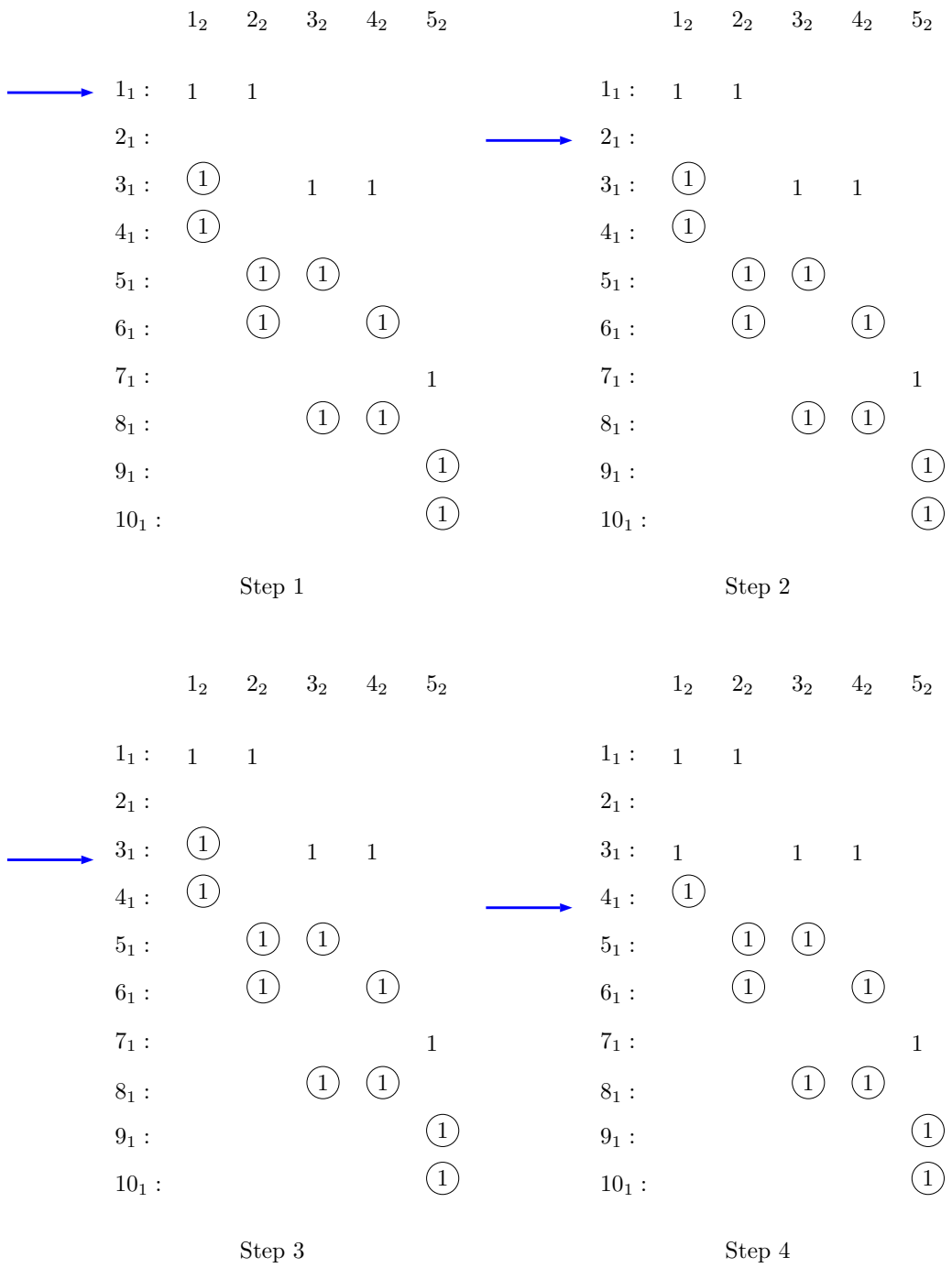
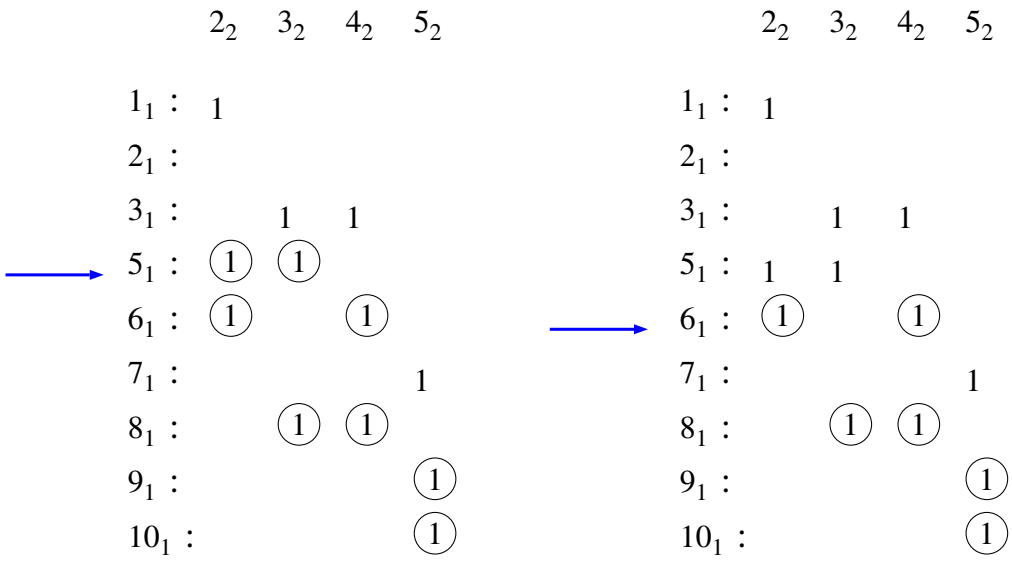
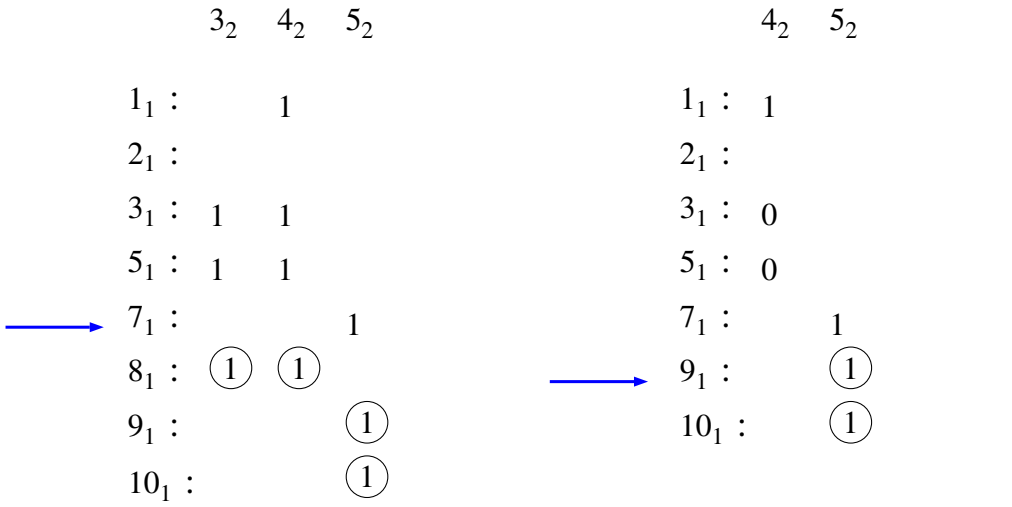


Fig. 4.6 Matching and V-path Procedure for ∂_2



Step 5 Step 6



Step 7 Step 8

Fig. 4.7 Matching and V-path Procedure for ∂_2 (continue)

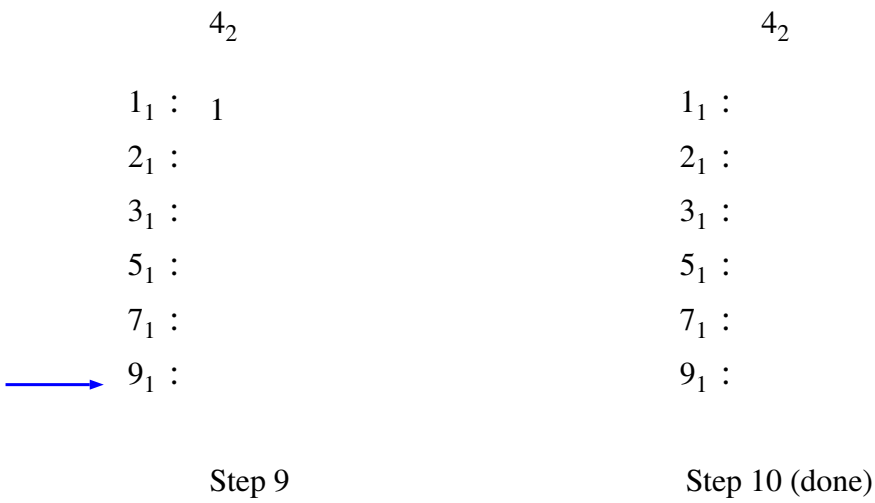


Fig. 4.8 Matching and V-path Procedure for ∂_2 (continue)

$$\begin{array}{cc} & 7_1 \\ \partial_1^{\mathscr{M}} \quad 1_0 : & 0 \end{array} \qquad \begin{array}{cc} & 4_2 \\ \partial_2^{\mathscr{M}} \quad 7_1 : & 0 \end{array}$$

Fig. 4.9 Combining the results of parallel algorithm on ∂_1 and ∂_2 in order to get the final $\partial_1^{\mathscr{M}}$ and $\partial_2^{\mathscr{M}}$

Chapter 5

Results

In this chapter, we are going to see how the proposed algorithms work on some simplicial complexes. We used the data sets from [3], where there is a library of triangulations. Here in order to check the complexity of algorithms for simplicial complexes of different dimensions, we measure three quantities: **real**, **user** and **system** time

1. **Real** time is the time from the invocation to the termination of a call.
2. **User** time is the amount of CPU time spent in user-mode code within the process.
3. **Sys** or system time is the amount of CPU time spent in the kernel within the process.

In general, real time is bigger than the total of user and sys time, because as explained in the definition, real time is the total elapsed time of a process from beginning till the end, while user and sys times show the time in a CPU. But, if we can split our process into some subprocesses (or threads) and the process is passed to more than one processors(or threads), then user and sys times can be even bigger than real time, because in this case they are equal to the sum of times in all processors and we are taking the advantage of parallel computing. We see the results in two parts: the first part is the benchmark for parallel matching in tables 3.3 and 3.4 and the second part will be based on homology computation based on discrete Morse theory that was explained in section 4.2.

Note that for a n-dimensional simplicial complex, we use the notation $f = (f_0, f_1, \cdots, f_n)$, where f_i represent the number of i-simplices.

5.1 Parallel Morse Matching

By looking at the discrete Morse vector in all tables, the first point is that for all simplicial complexes, the partial matching for ∂_1 is perfect, but when we go to higher dimensions, the matching will not be necessarily perfect. For example in figure 5.1, you see that for all different permutations on vertices, the discrete Morse vector is perfect. In some other simplicail complexes like 5.2, discrete Morse vector will not be perfect anymore and by different permutations on vertices and it may change.

Another issue is the complexity. When the dimension becomes greater than one, user time becomes greater than real time which means that we are taking the advantage of parallel computation. Real time tells us how much time gets an process, if it is not parallel. In figure 5.8, in the plot you see that when we increase the dimension, the gap between the real and user time will increase.

The disadvantage of this method is for higher dimensions, because we need to exclude some possibilities for making the parallel matching possible. One of the possible solutions for this problem is to try to combine this solution with Morse matching based on elementary collapses.

5.2 Results on Parallel Homology Calculation

In this section we concentrate on the practical results of algorithm that was explained in section 4.2. In figure 5.9, you see the complexity of the algorithm for some simplicial complexes. Like the previous section for Morse matching, you see when the dimension increases, the complexity does not increase linearly. In figure 5.10, you see the results for C.elegans that as we increase the dimension, the timing will not change and that is the advantage of parallelization.

The point that we have to mention here is that for homology based on matrix reduction, parallelization is possible, but for Morse matching it was not so and we had to do it sequentially. In this thesis, we proposed a method and made this paralleliza-

tion possible and by looking at diagrams in figure 5.11, you see that the performance with respect to the timing has been improved.

d2n12g6	Dimension $\leq$	Time			DMV
		real	user	sys	
P1	1	0.173s	0.115s	0.055s	$(1, 55)$
	2	0.249s	0.347s	0.128s	$(1, 12, 1)$
P2	1	0.160s	0.105s	0.055s	$(1, 55)$
	2	0.230s	0.287s	0.119s	$(1, 12, 1)$
P3	1	0.160s	0.105s	0.050s	$(1, 55)$
	2	0.230s	0.293s	0.119s	$(1, 12, 1)$
P4	1	0.160s	0.105s	0.050s	$(1, 55)$
	2	0.250s	0.307s	0.130s	$(1, 12, 1)$
P5	1	0.160s	0.105s	0.050s	$(1, 55)$
	2	0.256s	0.302s	0.142s	$(1, 12, 1)$
P6	1	0.160s	0.107s	0.051s	$(1, 55)$
	2	0.250s	0.305s	0.134s	$(1, 12, 1)$

Fig. 5.1 Results for different permutations of d2n12g6 with $f = (12, 66, 44)$ and optimal $DMV = (1, 12, 1)$

regular 2 21 23 1	Dimension $\leq$	Time			DMV
		real	user	sys	
P1	1	0.187s	0.131s	0.053s	$(1, 127)$
	2	0.64s	1.098s	0.124s	$(1, 36, 7)$
P2	1	0.191s	0.137s	0.05s	$(1, 127)$
	2	0.638s	1.94s	0.126s	$(1, 31, 2)$
P3	1	0.180s	0.127s	0.050s	$(1, 127)$
	2	0.63s	1.084s	0.124s	$(1, 30, 1)$
P4	1	0.180s	0.127s	0.049s	$(1, 127)$
	2	0.628s	1.074s	0.126s	$(1, 30, 1)$
P5	1	0.197s	0.133s	0.053s	$(1, 127)$
	2	0.63s	1.082s	0.125s	$(1, 30, 1)$
P6	1	0.186s	0.131s	0.051s	$(1, 127)$
	2	0.634s	1.084s	0.127s	$(1, 30, 1)$
P7	1	0.185s	0.129s	0.052s	$(1, 127)$
	2	0.633s	1.089s	0.124s	$(1, 30, 1)$

Fig. 5.2 Results for different permutations of regular 2 21 23 1 with $f = (21, 147, 98)$ and optimal $DMV = (1, 30, 1)$

rand2 n250 p0	Dimension $\leq$	Time			DMV
		real	user	sys	
P1	1	0.244s	0.190s	0.051s	$(1, 276)$
	2	20s	41s	0.163s	$(1, 12, 487)$
P2	1	0.292s	0.195s	0.053s	$(1, 276)$
	2	21s	42s	0.168s	$(1, 10, 485)$
P3	1	0.237s	0.172s	0.052s	$(1, 276)$
	2	21s	42s	0.175s	$(1, 7, 482)$
P4	1	0.236s	0.182s	0.051s	$(1, 276)$
	2	21s	41.7s	0.164s	$(1, 7, 482)$
P5	1	0.262s	0.197s	0.0513s	$(1, 276)$
	2	20.9s	41.5s	0.178s	$(1, 11, 486)$
P6	1	0.245s	0.189s	0.051s	$(1, 276)$
	2	20.5s	40.8s	0.174s	$(1, 9, 484)$
P7	1	0.248s	0.193s	0.052s	$(1, 276)$
	2	20.8s	41.3s	0.174s	$(1, 15, 490)$

Fig. 5.3 Results for different permutations of rand2 n250 p0 with $f = (25, 300, 751)$ and optimal $DMV = (1, 0, 475)$

Double Trefoil	Dimension $\leq$	Time			DMV
		real	user	sys	
P1	1	0.172s	0.117s	0.052s	$(1, 93)$
	2	0.9s	1.6s	0.156s	$(1, 3, 94)$
	3	1.15s	2.66s	0.442s	$(1, 3, 6, 4)$
P2	1	0.17s	0.115s	0.051s	$(1, 93)$
	2	0.835s	1.5s	0.124s	$(1, 5, 96)$
	3	1.1s	2.46s	0.356s	$(1, 5, 11, 7)$
P3	1	0.165s	0.113s	0.05s	$(1, 93)$
	2	0.8s	1.43s	0.115s	$(1, 5, 96)$
	3	1.1s	2.5s	0.375s	$(1, 5, 9, 5)$
P4	1	0.17s	0.115s	0.05s	$(1, 93)$
	2	0.82s	1.48s	0.117s	$(1, 3, 94)$
	3	1.2s	2.8s	0.45s	$(1, 3, 5, 3)$
P5	1	0.171s	0.116s	0.054s	$(1, 93)$
	2	0.83s	1.48s	0.123s	$(1, 3, 94)$
	3	1.2s	2.9s	0.413s	$(1, 3, 5, 3)$
P6	1	0.17s	0.115s	0.51	$(1, 93)$
	2	0.82s	1.5s	0.12s	$(1, 4, 95)$
	3	1.14s	2.64s	0.42	$(1, 4, 10, 7)$
P7	1	0.17s	0.115s	0.05s	$(1, 93)$
	2	0.8s	1.4s	0.123s	$(1, 5, 96)$
	3	1.05s	2.4s	0.38s	$(1, 5, 11, 7)$

Fig. 5.4 Results for different permutations of double trefoil data with $f = (16, 108, 184, 92)$ and optimal $DMV = (1, 0, 0, 1)$

Barnette spehere	Dimension $\leq$	Time			DMV
		real	user	sys	
P1	1	0.157s	0.103s	0.05s	$(1, 20)$
	2	0.216s	0.235s	0.126s	$(1, 0, 18)$
	3	0.271ss	0.441s	0.239s	$(1, 0, 0, 1)$
P2	1	0.17s	0.112s	0.055s	$(1, 20)$
	2	0.201s	0.223s	0.124s	$(1, 0, 18)$
	3	0.243s	0.418s	0.206s	$(1, 0, 0, 1)$
P3	1	0.157s	0.103s	0.05s	$(1, 20)$
	2	0.197s	0.222s	0.12s	$(1, 0, 18)$
	3	0.264s	0.452s	0.218s	$(1, 0, 0, 1)$
P4	1	0.162s	0.107s	0.051s	$(1, 20)$
	2	0.196s	0.219s	0.122s	$(1, 0, 18)$
	3	0.251s	0.411s	0.204s	$(1, 0, 0, 1)$
P5	1	0.171s	0.116s	0.054s	$(1, 20)$
	2	0.195s	0.219s	0.121s	$(1, 0, 18)$
	3	0.25s	0.416s	0.205s	$(1, 0, 01)$
P6	1	0.16s	0.105s	0.052s	$(1, 20)$
	2	0.199s	0.222s	0.12s	$(1, 0, 18)$
	3	0.266s	0.448s	0.238s	$(1, 0, 0, 1)$

Fig. 5.5 Results for different permutations of Barnette spehere data with $f = (8, 27, 38, 19)$ and optimal $DMV = (1, 0, 0, 1)$

Poincare	Dimension $\leq$	Time			DMV
		real	user	sys	
P1	1	0.182s	0.119s	0.051s	$(1, 91)$
	2	0.775s	1.35s	0.117s	$(1, 4, 93)$
	3	1.09s	2.53s	0.35s	$(1, 4, 8, 5)$
P2	1	0.194s	0.131s	0.056s	$(1, 91)$
	2	0.838s	1.5s	0.117s	$(1, 3, 92)$
	3	1.08s	2.57s	0.4s	$(1, 3, 7, 5)$
P3	1	0.168s	0.114s	0.05s	$(1, 91)$
	2	0.79s	1.39s	0.124s	$(1, 3, 92)$
	3	1.06s	2.44s	0.413s	$(1, 3, 5, 3)$
P4	1	0.173s	0.117s	0.053s	$(1, 91)$
	2	0.79s	1.38s	0.127s	$(1, 3, 92)$
	3	1.35s	2.73s	0.485s	$(1, 3, 7, 5)$
P5	1	0.183s	0.124s	0.054s	$(1, 91)$
	2	0.78s	1.39s	0.114s	$(1, 3, 92)$
	3	1.02s	2.32s	0.367s	$(1, 3, 8, 6)$
P6	1	0.17s	0.116s	0.05s	$(1, 91)$
	2	0.78s	1.39s	0.117s	$(1, 4, 93)$
	3	1.04s	2.28s	0.42s	$(1, 4, 6, 3)$

Fig. 5.6 Results for different permutations of poincare data with $f = (16, 106, 180, 90)$ and optimal $DMV = (1, 2, 2, 1)$

Non 4 2 Colorable	Dimension $\leq$	Time			DMV
		real	user	sys	
P1	1	2m0.5s	2m0.250	0.073s	$(1, 1413)$
	2	252m	504s	2.4s	$(1, 30, 1441)$
	3	315m	868m	2.5s	$(1, 30, 71, 42)$
P2	1	1m57s	1m56s	0.09s	$(1, 1413)$
	2	240m	480m	1.9s	$(1, 80, 1491)$
	3	310m	876m	2.6s	$(1, 80, 165, 86)$
P3	1	2m2.9s	2m1.6s	0.1s	$(1, 1413)$
	2	297m	587m	5.75s	$(1, 83, 1494)$
	3	362m	1014m	11.5s	$(1, 83, 168, 86)$
P4	1	2m28.72	2m28.40	0.08s	$(1, 1413)$
	2	283m	543m	2.8s	$(1, 77, 1488)$
	3	344m	950m	3.46s	$(1, 77, 168, 92)$
P5	1	1m59.51	1m59.20	0.07s	$(1, 1413)$
	2	252m	498m	2.2s	$(1, 84, 1495)$
	3	408m	1017m	5.7s	$(1, 84, 168, 85)$
P6	1	2m3ss	2m2.8s	0.07s	$(1, 1413)$
	2	305m	564m	8s	$(1, 56, 1467)$
	3	420m	1059m	17s	$(1, 56, 96, 41)$

Fig. 5.7 Results for different permutations of non 4 2 colorable data with $f = (167, 1579, 2824, 1412)$ and optimal $DMV = (1, 0, 0, 1)$

600 cell	Dimension $\leq$	Time			DMV
		real	user	sys	
P1	1	4.23s	4.08s	0.08s	$(1, 601)$
	2	3m33s	6m5s	2.4s	$(1, 0, 599)$
	3	4m34s	10m5s	1m25s	$(1, 0, 0, 1)$

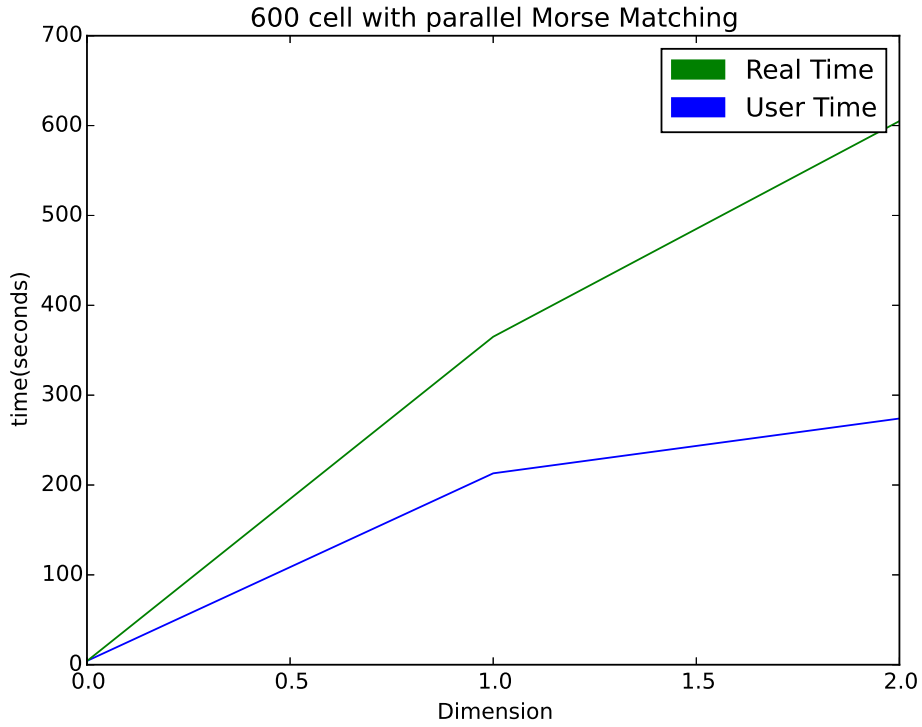


Fig. 5.8 Results for 600 cell with $f = (120, 720, 1200, 600)$ and optimal $DMV = (1, 0, 0, 1)$

Data	Dimension $\leq$	Time			Betti numbers
		real	user	sys	
Rudin	1	204	137	55	$(1, 53, 0, \cdots, 0)$
	2	348	484	140	$(1, 0, 41, 0, \cdots, 0)$
	3	383	769	264	$(1, 0, 0, 0, \cdots, 0)$
Trefoil	1	182	128	51	$(1, 57, 0, \cdots, 0)$
	2	301	439	115	$(1, 0, 55, 0, \cdots, 0)$
	3	460	930	269	$(1, 0, 0, 1, 0, \cdots, 0)$
Trefoil Arc	1	195	134	55	$(1, 47, 0, \cdots, 0)$
	2	293	403	128	$(1, 0, 38, 0, \cdots, 0)$
	3	354	672	229	$(1, 0, 0, 0, \cdots, 0)$
Poincare	1	195	138	54	$(1, 91, 0, \cdots, 0)$
	2	540	895	128	$(1, 0, 89, 0, \cdots, 0)$
	3	850	1885	434	$(1, 0, 0, 1, 0, \cdots, 0)$
Double Trefoil	1	200	138	54	$(1, 93, 0, \cdots, 0)$
	2	550	1925	130	$(1, 0, 91, 0, \cdots, 0)$
	3	870	1900	408	$(1, 0, 0, 1, 0, \cdots, 0)$
Triple Trefoil Arc	1	192	135	54	$(1, 111, 0, \cdots, 0)$
	2	724	1265	124	$(1, 0, 97, 0, \cdots, 0)$
	3	1070	2550	446	$(1, 0, 0, 0, \cdots, 0)$
Hyper.Dode.Space	1	207	147	56	$(1, 170, 0, \cdots, 0)$
	2	2000	3800	124	$(1, 0, 168, 0, \cdots, 0)$
	3	3614	8781	1116	$(1, 0, 0, 1, 0 \cdots, 0)$

Fig. 5.9 Complexity and Betti Numbers for different dimensions and data

Data	Dimension $\leq$	Time			Betti numbers
		real	user	sys	
Celegance	1	1m50s	1m49s	0.05s	$(1, 949, 0, \cdots, 0)$
	2	22m50s	45m27s	0.241s	$(1, 251, 187, 0, \cdots, 0)$
	3	23m	69m	0.3s	$(1, 251, 15, 0, \cdots, 0)$
	4	23m	93m5s	0.4s	$(1, 251, 15, 0, 0, \cdots, 0)$
	5	23m3.8s	94m56s	0.5s	$(1, 251, 15, 0, 0, \cdots, 0)$

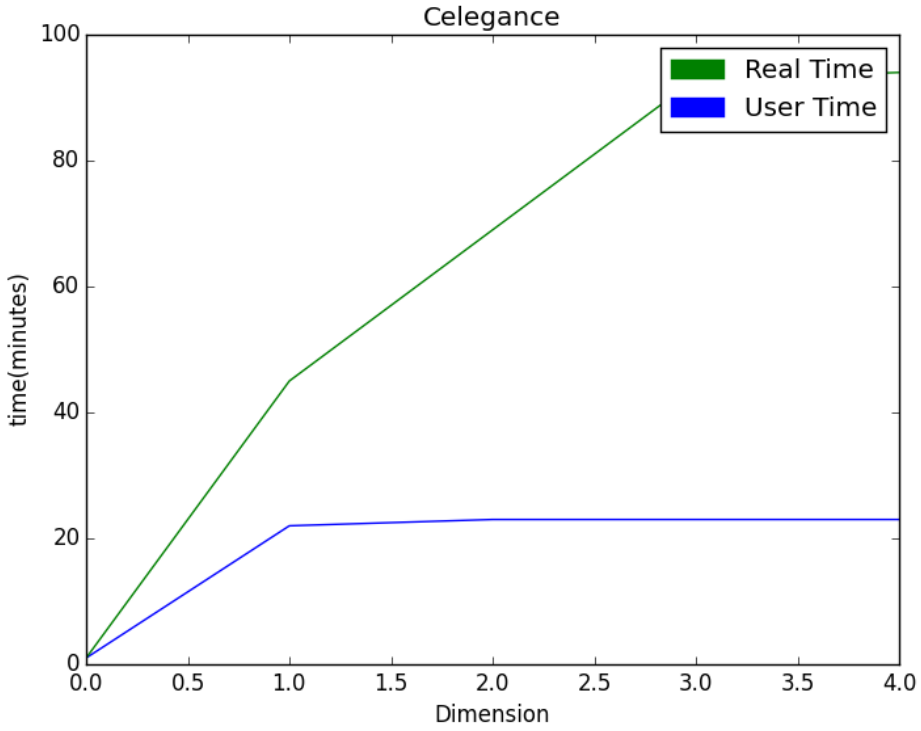


Fig. 5.10 Complexity and Betti Numbers for different dimensions and of C.elegans by Applying Parallel Morse Algorithm

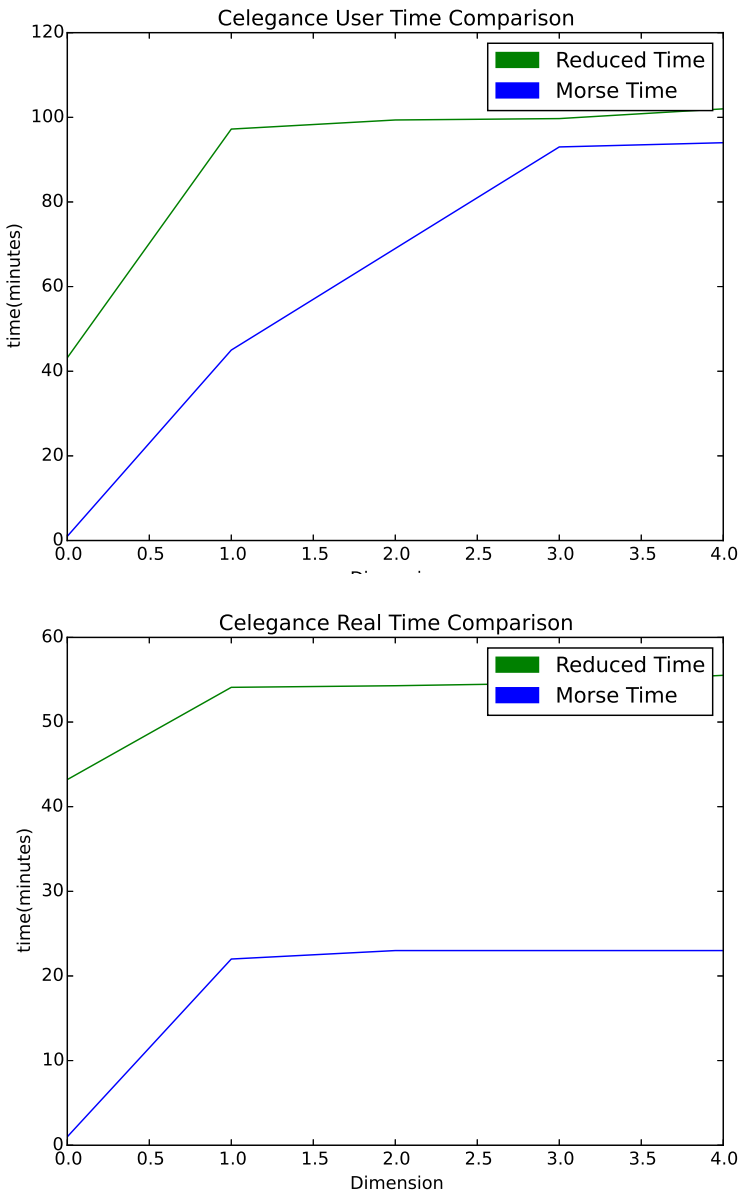


Fig. 5.11 Complexity comparison between method based on morse matching and method based on reducing the matrix

Data	Dimension $\leq$	Time			Betti numbers
		real	user	sys	
Hom C5 K4	1	2m15.8s	2m7.1s	2.55s	$(1, 1441, 0, \cdots, 0)$
	2	160m31s	313m27s	4.98s	$(1, 0, 1239, 0, \cdots, 0)$
	3	205m	603m	7.5s	$(1, 0, 0, 1, 0, \cdots, 0)$
Knot	1	5m25.3s	5m24.9s	0.095s	$(1, 1550, 0, \cdots, 0)$
	2	189m31s	370m27s	7s	$(1, 0, 1172, 0, \cdots, 0)$
	3	176m50s	530m	5.6s	$(1, 0, 0, 0, \cdots, 0)$
Non 4 2 Colorable	1	57s	56.8s	0.09s	$(1, 1413, 0, \cdots, 0)$
	2	126m	252m	1.6s	$(1, 0, 1412, 0, \cdots, 0)$
	3	186m	547m	3.5s	$(1, 0, 0, 1, \cdots, 0)$
600 cell	1	13.8s	13s	0.06s	$(1, 601, 0, \cdots, 0)$
	2	10m53s	21m42s	0.19s	$(1, 0, 599, 0, \cdots, 0)$
	3	12m33s	37m34s	0.3s	$(1, 0, 0, 1, 0, \cdots, 0)$

Fig. 5.12 Complexity and Betti Numbers for different dimensions and data (continue of table 5.9), these tests are done on polito super computer

Chapter 6

Conclusion

In this thesis, we proposed a parallel algorithm for finding a discrete Morse vector and actually it was a parallelized version of proposed algorithm in [17]. We just need to arrange the simplices in colexicographical order and as we proved, for up to two dimensional simplicial complexes, that worked perfectly and no double matching happens. But for higher dimensional simplicial complexes some double matching could happen, so by doing some modifications in the algorithm, we could fix this problem, even though the optimality of discrete Morse vector could be affected, but it could be a trade off between making the algorithm parallel and having a discrete Morse vector with minimal critical simplices.

Referring to the figures in the previous section, we see that due to parallelization by increasing the dimension, the complexity of algorithm does not necessarily increase. So, this method can be really useful for high dimensional simplicial complexes.

This idea can be extended for persistent homology as well and it can be done as a future work. Besides, here we worked on $\mathbb{Z}_2$ coefficients, and this methodology can be generalized and for example consider torsion as well.

References

[1] Y. Matsumoto. *An Introduction to Morse Theory*. Europe and Central Asia Poverty Reduction and Economic Manag. American Mathematical Society, 2002.

[2] Robin Forman. Morse theory for cell complexes. *Advances in Mathematics*, 134(1):90 – 145, 1998.

[3] Frank H Lutz Bruno Benedetti. Library of Triangulations. http://page.math.tu-berlin.de/~lutz/stellar/library_of_triangulations/.

[4] Robin Forman. A user's guide to discrete morse theory. *Séminaire Lotharingien de Combinatoire [electronic only]*, 48:B48c, 35 p., electronic only–B48c, 35 p., electronic only, 2002.

[5] N. Mramor Kosta, M. Pamuk, and H. Varli. Perfect Discrete Morse Functions on Connected Sums. *ArXiv e-prints*, January 2015.

[6] Robin Forman. Witten–morse theory for cell complexes†. *Topology*, 37(5):945 – 979, 1998.

[7] J.W. Milnor. *Morse Theory*. Annals of mathematics studies. Princeton University Press, 1963.

[8] K.P. Knudson. *Morse Theory: Smooth and Discrete*. World Scientific, 2015.

[9] Yanfeng Chen. A brief history of morse homology.

[10] Bruno Benedetti and Frank H. Lutz. Random discrete morse theory and a new library of triangulations. *CoRR*, abs/1303.6422, 2013.

[11] E.C. Zeeman. On the dunce hat. *Topology*, 2(4):341 – 358, 1963.

[12] Rafael Ayala, Desamparados Fernández-Ternero, and José Antonio Vilches. Perfect discrete morse functions on 2-complexes. *Pattern Recognition Letters*, 33(11):1495 – 1500, 2012. Computational Topology in Image Context.

[13] Michael Joswig and Marc E. Pfetsch. Computing optimal Morse matchings. *SIAM Journal on Discrete Mathematics*, 20(1):11–25, 2006.

[14] Alexander Engström. Discrete morse functions from fourier transforms. *Experimental Mathematics*, 18(1):45–53, 2009.

[15] Dmitry N. Kozlov. *Combinatorial Algebraic Topology*, volume 21 of *Algorithms and computation in mathematics*. Springer, 2008.

[16] Marian Mrozek and Bogdan Batko. Coreduction homology algorithm. *Discrete & Computational Geometry*, 41(1):96–118, 2009.

[17] P. Dłotko and H. Wagner. Computing homology and persistent homology using iterated Morse decomposition. *ArXiv e-prints*, October 2012.

[18] R. C. Prim. Shortest connection networks and some generalizations. *Bell System Technology Journal*, 36:1389–1401, 1957.

[19] Thomas Lewiner, Hélio Lopes, and Geovan Tavares. Optimal discrete morse functions for 2-manifolds. *Computational Geometry*, 26(3):221 – 233, 2003.

[20] Mark Newman. *Networks: An Introduction*. Oxford University Press, Inc., New York, NY, USA, 2010.

[21] A. Hatcher. *Algebraic Topology*. Algebraic Topology. Cambridge University Press, 2002.

[22] Afra J. Zomorodian, M. J. Ablowitz, S. H. Davis, E. J. Hinch, A. Iserles, J. Ockendon, and P. J. Olver. *Topology for Computing (Cambridge Monographs on Applied and Computational Mathematics)*. Cambridge University Press, New York, NY, USA, 2005.

Appendix A

Complexes

Following [21], we give some definitions.

Definition A.0.1 (k-cell)**.** *A k-dimensional cell or briefly k-cell is a topological space X homeomorphic to k-ball.*

Definition A.0.2 (Cell Complex)**.** *A cell or a CW complex is a space X constructed in the following way:*

1. *Start with a discrete set X^0, whose points are regarded as 0-cells.*
2. *Inductively, form the n-skeleton X^n from X^{n-1} by attracting n-cells e^n_α via maps $\phi^\alpha : S^{n-1} \to X^{n-1}$. This means that X^n is the quotient space of $X^{n-1} \sqcup_\alpha D^n_\alpha$ under the identifications $x \sim \phi_\alpha(x)$ for $x \in \partial D^n_\alpha$. The cell e^n_α is the homeomorphic image of $D^n_\alpha - \partial D^n_\alpha$ under the quotient map.*
3. *$X = \cup_n X^n$ with the weak topology.*

Definition A.0.3 (abstract simplicial)**.** *A finite abstract simplicial complex is a finite set S together with a collection K of subsets of S such that if $\sigma \in K$ and $\tau \subset \sigma$, then $\tau \in K$.*

Following [22], we present the following definitions.

Definition A.0.4 (k-simplex)**.** *A k-simplex is the convex hull of k+1 linearly independent set of points $S = \{v_0, v_1, \cdots, v_k\}$.*

Definition A.0.5 (face(coface))**.** *A face of σ is the convex hull τ of a non-empty subset of the v_i and we write $\tau \preceq \sigma$. Besides, it is proper, if the subset is not the entire set. If τ is the 1-codimensional face of σ, we write $\tau \lhd \sigma$ or $\sigma \rhd \tau$.*

Definition A.0.6 (simplicial complex)**.** *A simplicial complex K is a finite set of simplices such that*

1. $\sigma \in K, \tau \preceq \sigma \Rightarrow \tau \in K$
2. $\sigma, \sigma' \in K \Rightarrow \sigma \cap \sigma' \preceq \sigma, \sigma'$ *or* $\sigma \cap \sigma' = \emptyset$

Definition A.0.7. *[Facet] A facet is any simplex in a complex that is not a face of any larger simplex*

Definition A.0.8 (Pure Simplicial Complex)**.** *A simplicial complex K is called pure, if all of its facets have the same dimension.*

Appendix B

Some Notes on Set Theory

Here, we give two definitions for sets.

Definition B.0.1 (Lexicographical Order)**.** *Consider a set* $S = \{v_0, v_1, \cdots\}$*, where for each* $i < j$*,* $v_i < v_j$*. For two subset of* S*,* $A_1 = \{v_{i_0}, v_{i_1}, \cdots, v_{i_m}\}$ *and* $A_2 = \{v_{j_0}, v_{j_1}, \cdots, v_{j_m}\}$*, we say* $A_1 <_{lex} A_2$*, if:*

$$\exists\, k : \forall\, t < k \;\; v_{i_t} = v_{j_t}, v_{i_k} < v_{j_k}$$

Definition B.0.2 (Colexicographical Order)**.** *Consider a set* $S = \{v_0, v_1, \cdots\}$*, where for each* $i < j$*,* $v_i < v_j$*. For two subset of* S*,* $A_1 = \{v_{i_0}, v_{i_1}, \cdots, v_{i_m}\}$ *and* $A_2 = \{v_{j_0}, v_{j_1}, \cdots, v_{j_m}\}$*, we say* $A_1 <_{col} A_2$*, if:*

$$\exists\, k : \forall\, t > k \;\; v_{i_t} = v_{j_t}, v_{i_k} < v_{j_k}$$

In order to clarify the difference among these two ordering, let us give an example.

Example B.0.1. *Suppose* $S = \{v_0, v_1, v_2, v_3\}$ *where* $v_0 < v_1 < v_2 < v_3$*. Consider all subsets of* S *with 2 elements. In lexicographical order, the ordering is:*

$$\{v_0, v_1\} <_{lex} \{v_0, v_2\} <_{lex} \{v_0, v_3\} <_{lex} \{v_1, v_2\} <_{lex} \{v_1, v_3\} <_{lex} \{v_2, v_3\}$$

while in colexicographical order, we have:

$$\{v_0, v_1\} <_{col} \{v_0, v_2\} <_{col} \{v_1, v_2\} <_{col} \{v_0, v_3\} <_{col} \{v_1, v_3\} <_{col} \{v_2, v_3\}$$

Appendix C

Basic Concepts in Algebraic Topology

Following [22] and [21], we have the following definitions.

Definition C.0.1 (Homotopy)**.** *Consider mappings $f, g : X \longrightarrow Y$. We say f is **homotopic** to g if for all points $x \in X$, there exists a continuous map $F : X \times I \longrightarrow Y$ such that $F(x,0) = f(x)$ and $F(x,1) = g(x)$. The map F is called a **homotopy** from f to g.*

Definition C.0.2 (Homotopy Equivalence)**.** *Two spaces X and Y are **homotopy equivalent**, if there exist two maps $f : X \longrightarrow Y$ and $g : Y \longrightarrow X$ where $f \circ g$ is homotopic to identity map on Y and $g \circ f$ is homotopic to identity map on X.*

Definition C.0.3. *Let K be a simplicial complex. The **k'th chain group** of K denoted by $C_k(K)$, is a vector space spanned by k-simplices of K and the **k'th boundary operator** $\partial_k : C_k \to C_{k-1}$ such that:*

$$\partial_k[v_0, v_1, \cdots, v_k] = \Sigma_{j=1}^{k}(-1)^j[v_0, v_1, \cdots, \hat{v}_j, \cdots, v_k]$$

Then the kth homology group of a simplicial complex K is defined as:

$$H_k(K) = ker\,\partial_k / Im\,\partial_{k+1}$$

If the simplicial complex has dimension equal to n, then we have a chain of n boundary operations. Here, $ker\,\partial_k$ is called **kth cycle group** and it is shown by Z_k and $Im\,\partial_k$ is called **kth boundary group** and it is shown by B_k.

The dimension of the free part of H_k is called ***k*th Betti number**. If we write the kth homology group in the following form:

$$H_k(K) = D^{\beta} \oplus (\bigoplus_{i=1}^{n} D/d_i D)$$

then β here in this formula corresponds exactly to Betti number, because it's the dimension of the free part of the homology group.

In order to have a better understanding of Betti number consider a donut. For a donut $\beta_0 = 1$, because donut is a connected object, $\beta_1 = 2$, since it has two holes and finally $\beta_2 = 1$ for the void.

Definition C.0.4 (Euler Characteristics)**.** *Euler Characteristics $\mathcal{X}$ of a simplicial complex K is the alternating sum of Betti numbers, namely:*

$$\mathcal{X}(K) = \beta_0 - \beta_1 + \beta_2 + \cdots$$