Practical Considerations in Morse-Smale Complex Computation

Attila Gyulassy[1,2], Peer-Timo Bremer[2], Bernd Hamann[1], and Valerio Pascucci[3]

[1] Institute for Data Analysis and Visualization (IDAV),
Department of Computer Science, University of California, Davis, CA 95616
[2] Center for Applied Scientific Computing,
Lawrence Livermore National Laboratory
P.O. Box 808, L-561, Livermore, CA 94551
[3] Scientific Computing and Imaging Institute, University of Utah – School of Computing
72 S Central Campus Drive, 3750 WEB, Salt Lake City, UT 84112

Abstract. The Morse-Smale complex is an effective topology-based representation for identifying, ordering, and selectively removing features in scalar-valued data. Several algorithms are known for its effective computation, however, common problems pose practical challenges for any feature-finding approach using the Morse-Smale complex. We identify these problems and present practical solutions: (1) we identify the cause of spurious critical points due to simulation of simplicity, and present a general technique for solving it; (2) we improve simplification performance by reordering critical point cancellation operations and introducing an efficient data structure for storing the arcs of the complex; (3) we present a practical approach for handling boundary conditions.

1 Introduction

Scientific data is becoming increasingly complex, and sophisticated techniques are required for its effective analysis and visualization. The Morse-Smale (MS) complex is an efficient data structure that represents the complete gradient flow behavior of a scalar function, and can be used to identify, order, and selectively remove features. Although several algorithms are known for its computation, achieving an efficient implementation is still a challenge. Key optimizations remain un-addressed in the literature without which computation of the complex might have a memory footprint that grows past practical limits, simplification times that take days instead of seconds, and undesirable artifacts on the boundaries. We examine the causes of such problems and present the necessary techniques for overcoming them.

2 Related Work

The MS complex is a topological data structure that provides an abstract representation of the gradient flow behavior of a scalar field [12, 13]. Edelsbrunner et al. [3] defined the MS complex for piecewise-linear 2-manifolds by considering the PL function as the limit of a series of smooth functions, and used this interpretation to transfer ideas from

the smooth case. They also provided an efficient algorithm to compute the MS complex, restricted to edges of the input triangulation, and construct a hierarchical representation by repeated cancellation of pairs of critical points. Bremer et al. [1] improved the algorithm and described a multi-resolution representation of a scalar field. Both algorithms trace paths of steepest ascent and descent beginning at saddle points. These paths constitute boundaries of two-dimensional cells of the MS complex.

Cells in the MS complex of a trivariate scalar field can be of dimension zero, one, two, or three. Tracing boundaries of the three-dimensional cells while maintaining a combinatorial valid complex is a difficult task and a practical implementation of such an algorithm remains a challenge [2]. Nevertheless, the MS complex has been computed for trivariate scalar field data and successfully used to identify features through repeated application of atomic cancellation operations [7]. Computation of the complex in this manner makes necessary a preprocessing step that subdivides every voxel by inserting "dummy" critical points, and therefore has a large computational overhead. This approach was improved by using a sweeping plane [8], but data size and computational overhead still turned out to be a limiting factor. An algorithm based on region-growing [9] was introduced for simplicial meshes of three dimensions, with a tenfold improvement in efficiency, however, the need to store the ascending and descending manifold membership information at each cell of the input, and the requirement to represent the entire output explicitly limits the scalability of this approach.

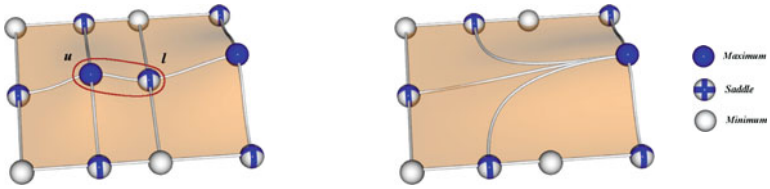
Discrete Morse theory, as presented by Forman [5], has also been used to compute the MS complex. Lewiner et al. [11] showed how a discrete gradient field can be constructed and used to identify the MS complex, however, this construction requires modification of the input mesh and an explicit representation of gradient paths, restricting the applicability of the method. King et al. [10] presented a method for constructing a discrete gradient field that agrees with the large-scale flow behavior of the data defined at vertices of the input mesh. Gyulassy et al. [6] presented an algorithm to compute the MS complex for data of any dimension in a memory-efficient manner by subdividing the data into parcels, computing the discrete gradient and complex on each parcel, and gluing the complex of each parcel back together using the discrete gradient flow across the boundaries. This method was the first scalable algorithm published for constructing MS complexes for large data. We use the slicing version of this algorithm to generate memory and run-time statistics is this paper.

3 Background

We present some basic definitions and an explanation of cancellations in the Morse-Smale complex.

3.1 Morse Functions and the Morse-Smale Complex

A real-valued smooth map $f : \mathbb{M} \to \mathbb{R}$ defined over a compact d-manifold $\mathbb{M}$ is a *Morse function* if all its critical points are non-degenerate (*i.e.*, the Hessian matrix is non-singular for all critical points) and no two critical points have the same function value. An integral line of f is a maximal path in $\mathbb{M}$ whose tangent vectors agree with the



(a) (b)

Fig. 1. The circled arc connects a saddle l to a maximum u (a). cancellation of (l,u) removes all arcs attached to l or u, and creates new arcs from the lower neighbors of u to the upper neighbors of l (b).

gradient of f at every point of the path. Each integral line has a natural origin and destination at critical points of f where the gradient becomes zero. *Ascending* and *descending* manifolds are obtained as clusters of integral lines having common origin and destination respectively. The *Morse-Smale (MS) complex*, denoted Γ, is a partition of $\mathbb{M}$ into regions clustering integral lines that share common origin and destination. In Morse-Smale functions, the integral lines only connect critical points of different indices.

Each critical point of index n is the origin of a set of integral lines that forms an ascending $d-n$-manifold. Symmetrically, it is the destination of a set of integral lines that forms a descending n-manifold. All ascending and descending manifolds of a Morse-Smale function intersect transversely. Therefore, given two critical points a and b, where the index of a is one less than the index of b, the intersection of the ascending manifold of a and the descending manifold of b is either empty or a 1-manifold. The critical points and these 1-manifolds are called *nodes* and *arcs*. The one-skeleton formed by the nodes and arcs forms the *combinatorial structure* of the MS complex. The combinatorial structure contains much of the semantic information of f, and is useful for simplification and feature identification. The *neighborhood* of a node a of an MS complex Γ is the set of nodes N_a that are connected to a by an arc in Γ.

3.2 Cancellations

A function f is simplified by repeated cancellation of pairs of critical points. The local change in the MS complex leads to a smoothing of the gradient vector field and hence of the function f. A cancellation operation is *valid* (*i.e.*, it can be realized by a local perturbation of the gradient vector field) for a pair of critical points if and only if there exists exactly one arc connecting them in the complex. Therefore, the indices of the two critical points must differ by one. Also, any critical point pair that is connected by multiple arcs represents a configuration known as a *strangulation* or a *pouch*, for which there is no direct perturbation of the gradient that removes the critical point pair. The atomic cancellation operations that are the basis for simplifying an MS complex were characterized in Gyulassy et al. [6] for complexes of general dimensions:

Cancellation: Let Γ be an MS complex for a scalar function defined on a closed d-manifold $\mathbb{M}$. Let l and u be the lower and upper nodes of an arc a in Γ, with indices i

and $i+1$, respectively. Let A_l be the set of arcs that have l as one end point, A_u the set of arcs that have u as one end point, N_l the set of nodes in the neighborhood of l, and N_u the set of nodes in the neighborhood of u.

The *combinatorial cancellation* of (l,u) changes the combinatorial structure of the MS complex and is based on these two steps:

1. Creation of a new arc connecting every critical point of indices $i+1$ in N_l to every critical point of index i in N_u, and adding them to Γ.
2. Removal of arcs in A_l and A_u, and removal of l and u from the complex.

Figure 1 shows this operation for a two-dimensional MS complex. The cancellation operation creates and destroys several arcs in the MS complex, and therefore an efficient data structure representing the nodes and arcs is necessary to handle the operations involved. Let $a=(l,u)$ be an arc that is canceled, n be the number of critical points of index $i+1$ in N_l, and m be the number of critical points of index i in N_u. In the fist step of canceling a, $n \times m$ new arcs are added to the MS complex. Each arc that is created must be inserted into the set of arcs of the nodes at its endpoints, resulting in $O(n \times m)$ INSERT operations. In the second step, the arcs connecting with the canceled nodes are removed from the complex. Each arc that is removed from the complex must be deleted from the set of arcs of the nodes at its endpoints, resulting in $O(|A_l|+|A_u|)$ DELETE operations. Therefore, a data structure for storing the arcs at each node must support efficient INSERT and DELETE operations.

In previous implementations [6,7,9], all arcs connected to a node were stored in a linked list. INSERT operations were performed in constant time, however, the DELETE operations were linear time in the number of arcs $|A_n|$ connected to a node n. Therefore, the total running time for a cancellation was $O((n \times m)+((|A_l|+|A_u|) \times A_m))$ where m is the node with the largest number $|A_m|$ of connected arcs in $N_u \cup N_l$.

The first term in this running time can lead to a quadratic increase in the number of arcs in the complex: not only does it create $n \times m$ new arcs, but every index-i node in N_u has m arcs added to its set of connected nodes, and every index-$(i+1)$ node in N_l has n arcs added. Therefore, future cancellations between pairs of these nodes become more costly. Therefore, performing k cancellations in a naive ordering can therefore lead to an actual $kn \times km$ cost. Such situations often arise when large flat regions have many adjacent 0-persistence arcs, as is the case in most integer-valued data. The second term in the running time can also be prohibitive, as nodes with a large number of connected arcs (high valence) slow the cancellation of every node that is connected to it. In fact, a linear-time DELETE operation in the number of arcs connected to a node leads to quadratic-time cancellations in the number of arcs connected to each endpoint.

3.3 Data Sets

The timing and memory performance statistics presented throughout this paper were generated on an off-the-shelf 2.21GHz AMD Athlon processor with 2.0Gb memory. Data sets were chosen to stress particular aspects of the MS complex construction algorithm. The data sets used and the particular challenge posed by each are summarized in table 1.

Name	Size	Values	Description	Challenge
Artificial	$64 \times 64 \times 64$	Byte	An artificially generated data set where every cell is critical and all arcs have zero persistence	No implicit ordering of arcs for cancellation
Hydrogen	$128 \times 128 \times 128$	Byte	The spatial probability distribution of an electron around a hydrogen atom in a strong magnetic field	Large flat regions with simple persistent features
Aneurism	$256 \times 256 \times 256$	Byte	A rotational X-ray scan of an aneurism	Flat regions interspersed with noise
Porous	$230 \times 230 \times 325$	Float	Distance to an interface surface in a simulated porous solid	Noisy floating-point data with flat ridge lines

Table 1. The data sets used for the generation of the timing data presented in this paper were selected to examine particular aspects of performance.

4 Improved Simulation of Simplicity

An interpolated function derived from real-life data is not necessarily Morse-Smale. One particular property necessary for algorithms computing the MS complex is differentiability, *i.e.*, vertices have pairwise distinct value, and hence no "flat" regions exist. Edelsbrunner and Mücke [4] introduced simulation of simplicity to resolve degenerate conditions where the input function is not a Morse function. Differentiability is simulated by assigning a strictly increasing ordering to vertices. Typically this is achieved with a pre-sort of the data, and vertex a has lower value than a vertex b if and only if it has lower index in the sorted array. In this manner, the order given by distinct data values is preserved, and an ordering for same-valued vertices is created. This sorting can be done implicitly, as shown in figure 2.

While this simple simulation of simplicity breaks ties and provides a complete ordering of the vertices of a data set, it introduces critical points that only exist due to index comparison. Figure 2 illustrates a scalar function with flat regions, where such an ordering produces spurious critical points at the boundaries of the flat regions. Although such critical points can be detected as noise and removed through simplification, they introduce a significant processing and data size overhead when computing the MS complex.

We improve the standard simulation of simplicity by introducing a new sorting order for the vertices that uses a greedy region-growing approach to eliminate the extra critical points due to flat regions. Our technique uses two priority queues to create a sorting order: the first, *simpleOrder*, returns the lowest unprocessed element ordered by function value and index in the data structure; the second, $bfsOrder$, has the same ordering, but only contains unprocessed vertices that are neighboring previously processed vertices. When a query asks for the lowest unprocessed vertex, the top elements of *simpleOrder* and $bfsOrder$ are compared, and the one with lower function value is returned. If they have the same function value, the top element of $bfsOrder$ is returned. When a vertex is processed, all of its unprocessed neighbors are added to $bfsOrder$.

In this manner, the ordering corresponds to the insertion time of a vertex in a flood-fill of the ascending manifolds of a function. It is a greedy technique that crosses flat

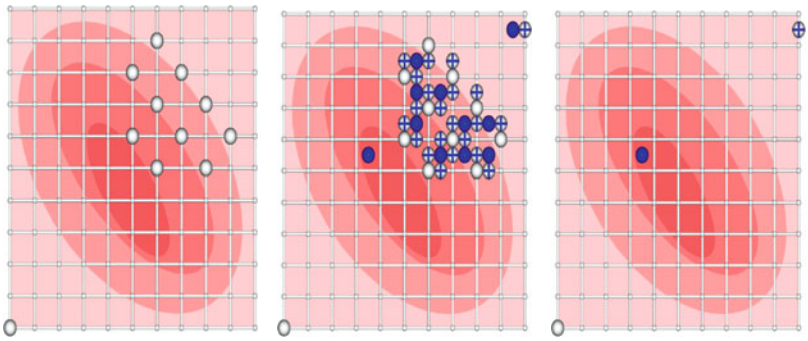


Fig. 2. Let the *index* of a vertex $v = (x, y)$ be $x + (y * xdim)$. Using standard simulation of simplicity, minima are identified at the boundaries of flat regions (a). When constructing an MS complex, these minima further generate the necessary separating saddles and maxima (b), far more than the actual three persistent critical points (c).

regions in a breadth-first search, and therefore prevents extra minima from appearing. Note that it is not necessary to build the sorting order explicitly when using an algorithm based on region growing, since often these algorithms only use the ordering to extract the lowest unprocessed element. This same ordering is particularly effective for ordering cells in the computation of a discrete gradient field. For example, in the algorithm presented by Gyulassy et al. [6], cells of each dimension d are iterated, and local minima correspond to critical cells of index d. The performance of this algorithm, number of critical points found, and memory footprint are summarized in Table 5. Although this method was primarily designed to help overcome the flat regions in integer-valued data, a large improvement was also seen in the porous data set, a floating-point data set. This behavior indicates that discretizing a function makes it very sensitive to the ordering of cells, even when the original values are reasonably distinct floating-point numbers.

5 Efficient Cancellations

We use the description of the atomic cancellation operations provided in Gyulassy et al. [6] as a basis for examining the necessary major improvements to enable simplification of the MS complex in practice. We increase the performance of cancellations by introducing a data structure for constant-time DELETE operations, and also present several strategies for preventing a dramatic increase in the number of arcs due to the potential $kn \times km$ new arcs created in a saddle-saddle cancellation.

5.1 Efficient Data Structure

We use a double-linked list to store the set of arcs A_n connected to a node n. The INSERT operation can be done in constant time, since new arcs are always inserted at the beginning of the list. A DELETE operation in a linked list requires finding the element

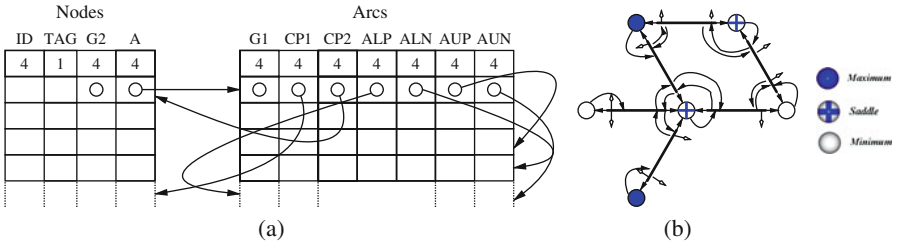


Fig. 3. The nodes and arcs of the complex are stored as fixed-size elements in an array (a) with the arcs containing the link information of the arc lists of their lower and upper nodes. ALN and ALP are the indices of the next and previous arcs in the double-linked arc list of the lower node, and AUN and AUP are the indices of the next and previous arcs in the list of the upper node. Each arc additionally has a pointer to its lower and upper nodes, and a pointer to the geometry of the arc. (b) shows a sample two-dimensional complex with the pointers at each node and arc.

Data set	Artificial	Hydrogen	Aneurism	Porous
(a)	620	951	638	3224
(b)	5130	5426	5006	5510

Table 2. A chart showing the sustained cancellation rate to cancel arcs 20% of maximum persistence for various data sets. (a) lists the number of cancellations per second using a standard linked list, and (b) shows the cancellations/second using a double-linked list. Note that the porous data set has a high rate of cancellations even when using a linked list, because it has very few high-valence nodes.

to remove, the previous element, and the next element, and reconnecting to remove the element. Finding an element is linear in the size of the list, while reconnecting is constant. A key observation is that in a cancellation operation, finding an arc in A_n is done in constant time, since the arc element is previously identified during an iteration of the arcs of its other endpoint. The previous and next elements must still be found, and in a double-linked list this is done in constant time. Therefore, the entire DELETE of an arc takes constant time in a cancellation operation. This behavior represents an improvement over the linear time for the linked list used in previous implementations [6].

Figure 3 illustrates the data structure used to store the nodes and arcs of the MS complex. Although this data structure is 1.4 times larger than a single-linked list, the majority of the memory footprint of a complex is taken up by the geometry components, such as storing the set of line segments that define an arc. In fact, for a typical complex, the additional space required by the arcs to store the double-linked list is only 0.1%. Table 2 compares the performance of using a standard linked list versus a double-linked list.

5.2 Cancellation Strategy

simplification of a function is achieved by repeated cancellation of critical point pairs. The ordering of critical point pairs is defined by *persistence*, which quantifies the importance of the topological feature associated with a pair. The *persistence* of a critical

point pair is the absolute difference in value of f between the two points. Typically, in this ordering, the arc with the lowest persistence is canceled first. However, such an ordering can lead to a memory footprint that grows past practical limits, due to the creation of $kn \times km$ new arcs. Instead, we design a strategy to determine when a cancellation can be delayed, based on conditions imposed on the neighborhood of the critical points to be canceled. We still maintain the property that all cancellations must be valid, *i.e.*, the pair of nodes is connected by exactly one arc, and the arc to be canceled has the lowest persistence of any in the neighborhood around the two nodes. Note that we do not follow a strict global persistence ordering, since there are small features that will not be removed until a later point in the simplification.

Limit $n \times m$ A straightforward technique to prevent the dramatic increase in memory is to delay a cancellation until the number of new arcs it would create, $n \times m$, is below some threshold t. Arcs are initially ordered by persistence. However, if canceling the arc with lowest persistence were to create more than t new arcs, the arc can be removed from the ordering and put in a delayed list, and the next arc in the ordering is selected for cancellation. cancellation of arcs in the neighborhood of delayed arcs can reduce the number of arcs incident at the nodes at either endpoint. Delayed arcs are canceled as soon as they create fewer than t new arcs. Table 3 shows the memory footprint and running times for several data sets as a function of t. While lower t values generally improve performance, they can lead to artifacts in the form of low persistence arcs that are perpetually delayed. When t is too high, the memory cost and run time increases, sometimes preventing the termination of the algorithm. However, t should be chosen as high as possible to keep the number of artifacts small.

Global Valence Control The *valence* of a node is the number of arcs that have it as an endpoint. Nodes with a high valence are expensive to cancel, and tend to generate many new arcs, leading to more high-valence nodes. We present a strategy to prevent the creation of high-valence nodes by delaying the cancellation of arcs that lead to the creation of a high-valence node. When an arc is canceled, the valence of nodes in a neighborhood around the canceled pair changes. Let n be the number of index-$(i+1)$ nodes in the neighborhood N_l of the lower node, m be the number of index-i nodes in the neighborhood N_u of the upper node. Let $a_i \in N_l$ be a node in the neighborhood of the lower node, and $b_i \in N_u$ be a node in the neighborhood of the upper node, and define $V(a)$ to return the valence of node a. We define the weight of an arc as $W(a) = max(max(V(a_i)+m|a_i \in N_l), max(V(b_i)+n|b_i \in N_u))$. Intuitively this value is the largest valence that canceling a would create. A cancellation is delayed if the weight of the arc is greater than a threshold t. Table 4 shows the memory footprint and running times for several datasets as a function of t. As with the new arc limiting method, lower t values correspond to higher performance and an increased number of artifacts. Computing $V(a_i)$ requires storage of the valence at each node, and an update of it whenever the neighborhood changes.

Data Set	t=20	t=40	t=80	t=160	t=360	t=720
Artificial-Time	–	6m 34s	12m 40s	11m 51s	25m 31s	–
Artificial-#Arc	–	13,054,772	14,392,886	14,398,002	25,652,629	–
Artificial-Atf.	–	887,281	145,332	24,045	10,244	–
Hydrogen-Time	0.86s	1.20s	2.03s	2.45s	2.63	3.30s
Hydrogen-#Arc	34,997	35,670	42,005	42,325	45,860	52,483
Hydrogen-Atf.	186	4	0	0	0	0
Aneurism-Time	–	1m 35s	4m 59s	8m 10s	16m 45s	47m 32s
Aneurism-#Arc	–	1,971,735	2,450,350	4,508,327	7,616,923	12,320,101
Aneurism-Atf.	–	23,320	1,625	122	34	6
Porous-Time	6m 25s	7m 04s	9m 32s	15m 45s	17m 16s	34m 47
Porous-#Arc	6,202,863	6,402,892	8,300,210	9,224,373	12,535,309	35,541,770
Porous-Atf.	352	202	54	17	6	5

Table 3. For each data set, we provide as a function of t: the time required to cancel up to 20% persistence, the maximum number of arcs in the complex, and the number of low-persistence artifacts that could not be cancelled, delaying arcs whose cancellation would generate more than $n \times m$ new arcs. In general, a higher threshold results in longer run times, a larger memory footprint, but fewer artifacts. Blanks indicate non-termination. The MS complexes were generated using the improved simulation of simplicity from Sect. 5.

Data Set	t=10	t=20	t=40	t=60	t=80
Artificial-Time	5m 20s	7m 35s	12m 40s	11m 51s	30m 52s
Artificial-#Arc	12,409,860	13,003,472	15,492,382	16,255,012	25,652,629
Artificial-Atf.	724,381	45,351	15,045	1,244	
Hydrogen-Time	1.06s	1.25s	2.18s	3.06s	3.54s
Hydrogen-#Arc	35,206	37,288	43,110	47,757	52,123
Hydrogen-Atf.	72	0	0	0	0
Aneurism-Time	1m 32s	2m 15s	10m 21s	17m 45s	25m 26s
Aneurism-#Arc	1,870,930	1,930,945	4,603,157	8,142,825	11,242,050
Aneurism-Atf.	26,076	18,450	102	3	2
Porous-Time	6m 05s	8m 36s	12m 29s	15m 54s	19m 24s
Porous-#Arc	5,730,043	7,404,100	10,004,540	13,224,373	15,853,612
Porous-Atf.	266	65	23	6	4

Table 4. For each data set, we provide as a function of t: the time required to cancel up to 20% persistence, the maximum number of arcs in the complex, and the number of low-persistence artifacts that could not be cancelled, delaying arcs whose cancellation would produce a node of valence greater than t.

Data set	Hydrogen	Aneurism	Porous
Standard Simp.	846,784	2,661,251	13,172,740
BFS Simplicity	3571	180,267	1,074,734

Table 5. The total number of critical points identified using a slice-by-slice computation of the discrete gradient. The top row lists the numbers of critical points found when the standard simulation of simplicity is used to order cells. The bottom row shows the numbers of critical points identified with the on-the-fly ordering.

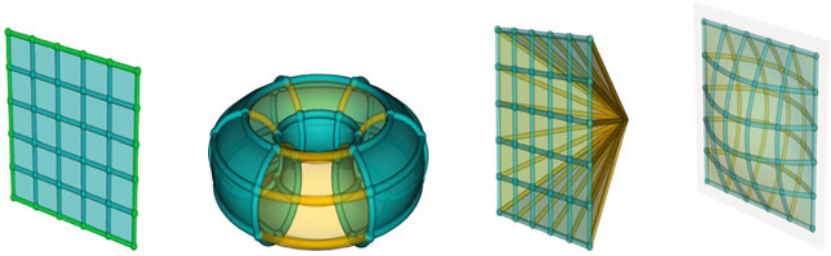
6 Boundary Conditions

The MS complex is well-defined for scalar functions on manifolds without boundary. Boundaries of a manifold present problems in computation of the MS complex, since they require special rules for the identification and simplification of critical points and cells. In practice, it is much easier to simulate the domain being a manifold without boundary, avoiding the special boundary cases. When the data is periodic, the underlying manifold has no boundary, and the MS complex is well-defined. When the data is not periodic, some other interpretation of the manifold must be used. Two standard techniques for removing the boundary of a manifold are one-point compactification and mirroring. Figure 4 illustrates how each boundary condition implies a different underlying space.

In one-point compactification, a vertex with infinite persistence is added for every boundary component, with every boundary cell attached to its corresponding vertex. While this can be simulated in practice, it can lead to artifacts, such as minima on the boundary disappearing and 1-saddles on the interior have arcs that connect outside the data. Additionally, the simulated vertices will necessarily have high valence, leading to slower execution times, and often require special data structures.

In mirrored boundary conditions, a copy of the underlying manifold is attached to itself along its boundary cells. This technique has the advantage of not introducing high-valence nodes, and small features near the boundary are preserved. However, it is often impractical to keep a redundant copy of the "mirrored" complex due to memory limitations. One solution is to modify the data structure of cells crossing the boundary, since differences between the MS complex on the original manifold and the complex on the mirrored manifold will be restricted to a neighborhood around the boundary. The biggest problem with this technique is that cancellation operations (in particular saddle-saddle cancellations) are not necessarily symmetric, therefore, canceling a node on the boundary with a node in the interior can lead to inconsistencies between the complex of the original manifold and the mirrored complex, and the technique loses its intuitive appeal.

A simple solution is possible in practice. We avoid picking a particular technique to remove the boundary by enforcing the condition that gradient flow not cross the boundary. This is done by computing the discrete gradient first on the boundary, then the interior of the domain. Additionally, when canceling critical points, boundary nodes are only canceled with other boundary nodes, and interior nodes with interior nodes. Therefore, the flow touching the boundary from the interior will always end at a critical point on the boundary, and flow touching the boundary from "outside" will end at a boundary critical point, effectively isolating the interior of the data. One major advantage of this technique is that it requires no special conditions for finding critical points and cells of the MS complex. The disadvantage is that small features on the boundary are often preserved, since they cannot be canceled with interior critical points. In practice, this is an effective technique that works for most cases. Any boundary artifacts can usually be removed from consideration by careful filtering of the MS complex.



(a) (b) (c) (d)

Fig. 4. A 2-manifold with interior (blue) and boundary (green) cells (a). Periodic boundary conditions (b) attach the boundary on opposing sides. One-point compactification (c) attaches the boundary to a point with infinite persistence. Mirrored boundary (d) duplicates the interior and attaches it along the boundary.

7 Conclusions

We present techniques that make computing and simplifying MS complexes possible in practice. Using the improved simulation of simplicity results in a more efficient implementation, especially in data sets with large flat regions, since it reduces the initial number of critical points found. In the case of the hydrogen dataset, the improved simulation of simplicity generated 200 times fewer critical cells than the standard simulation of simplicity. While particularly designed for computing MS complexes using a discrete approach, this simulation of simplicity can be applied to any topology-based technique to reduce the initial number of low-persistence critical points found. The techniques presented for reordering cancellations make it possible to simplify an MS complex without entering a situation where unregulated cancellations lead to memory usage that exceeds practical capabilities. The double-linked list we use to store the connectivity of the 1-skeleton of the MS complex further provides a decrease in run-time complexity of cancellation operations. Finally, we present a simple solution to enable a simple implementation when boundaries are present.

The techniques we have presented make possible computation and simplification of the MS complex. For example, without the improved simulation of simplicity, the MS complex of the hydrogen data set has almost one million low-persistence critical points. simplification of this large complex does not terminate without using a re-ordering strategy for cancellations. The hydrogen data set is a small data set with fewer than 100 persistent critical points, and the MS complex can be computed and simplified in under a minute when using the techniques discussed in this paper. These techniques are simple solutions to practical problems in designing efficient implementations of algorithms to compute and simplify the MS complex.

8 Acknowledgments

Attila Gyulassy was supported by the Lawrence Scholar Program (LSP). In addition, this research was supported in part by the National Science Foundation, under grant CCF-0702817. We would like to thank the members of the Center for Applied Scientific

Computing (CASC), at LLNL, and the members of the Visualization and Computer Graphics Research Group of the Institute for Data Analysis and Visualization (IDAV), at UC Davis. This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

References

1. P.-T. Bremer, H. Edelsbrunner, B. Hamann, and V. Pascucci. A topological hierarchy for functions on triangulated surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 10(4):385–396, 2004.
2. H. Edelsbrunner, J. Harer, V. Natarajan, and V. Pascucci. Morse-Smale complexes for piecewise linear 3-manifolds. In *Proc. 19th Ann. Sympos. Comput. Geom.*, pages 361–370, 2003.
3. H. Edelsbrunner, J. Harer, and A. Zomorodian. Hierarchical Morse-Smale complexes for piecewise linear 2-manifolds. *Discrete and Computational Geometry*, 30(1):87–107, 2003.
4. H. Edelsbrunner and E. P. Mücke. Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms. *ACM Trans. Graph.*, 9(1):66–104, 1990.
5. R. Forman. A user's guide to discrete morse theory, 2001.
6. A. Gyulassy, P.-T. Bremer, B. Hamann, and V. Pascucci. A practical approach to morse-smale complex computation: Scalability and generality. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1619–1626, 2008.
7. A. Gyulassy, V. Natarajan, V. Pascucci, P.-T. Bremer, and B. Hamann. Topology-based simplification for feature extraction from 3d scalar fields. In *Proc. IEEE Conf. Visualization*, pages 535–542, 2005.
8. A. Gyulassy, V. Natarajan, V. Pascucci, P. T. Bremer, and B. Hamann. A topological approach to simplification of three-dimensional scalar fields. *IEEE Transactions on Visualization and Computer Graphics (special issue IEEE Visualization 2005)*, pages 474–484, 2006.
9. A. Gyulassy, V. Natarajan, V. Pascucci, and B. Hamann. Efficient computation of morse-smale complexes for three-dimensional scalar functions. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1440–1447, 2007.
10. H. King, K. Knudson, and N. Mramor. Generating discrete morse functions from point data. *Experimental Mathematics*, 14(4):435–444, 2005.
11. T. Lewiner, H. Lopes, and G. Tavares. Applications of forman's discrete morse theory to topology visualization and mesh compression. *IEEE Transactions on Visualization and Computer Graphics*, 10(5):499–508, 2004.
12. S. Smale. Generalized Poincaré's conjecture in dimensions greater than four. *Ann. of Math.*, 74:391–406, 1961.
13. S. Smale. On gradient dynamical systems. *Ann. of Math.*, 74:199–206, 1961.