A KDS for discrete Morse-Smale complexes

A KDS for Discrete Morse-Smale Complexes*

T. Ophelders, W. Sonke, B. Speckmann, and K. Verbeek
Dep. of Mathematics and Computer Science, TU Eindhoven, The Netherlands
[t.a.e.ophelders|w.m.sonke|b.speckmann|k.a.b.verbeek]@tue.nl

Introduction. Consider a two-dimensional surface S with a height function $h : S \to \mathbb{R}$. The Morse-Smale complex (MS-complex) of T is a topological complex that provides information about the features of the height function on the terrain. It consists of the critical points (minima, saddles and maxima) of h in T, together with steepest-descent paths from saddles to minima and steepest-ascent paths from saddles to maxima. In the continuous case, the MS-complex is well-defined if h is a Morse function: each critical point of h has a distinct height, and certain types of degeneracies do not occur. To allow computing MS-complexes on real-world measurement data, which typically is discrete, several extensions of Morse functions to the discrete case have been studied. An extensive explanation of the most prominent of those, *discrete Morse theory*, is provided by Forman [2]. Based on discrete Morse theory, there have been several approaches to define discrete MS-complexes. In this work we focus on the discrete MS-complex defined by Shivashankar *et al.* [3]. We present a kinetic data structure (KDS) for this MS-complex. That is, we consider a height function h that changes over time, and provide a data structure to track the MS-complex throughout this movement. This can be used to efficiently analyze time-varying data.

Discrete MS-complex. The discrete MS-complex computed by Shivashankar *et al.* is defined by a discrete gradient field, which is a set of *gradient pairs*. While gradient fields are defined for any cell complex, to simplify the presentation, we assume here that the input is a triangulated (two-dimensional) terrain K. In this setting, there are two types of gradient pairs: those between vertices and edges, and those between edges and faces. Specifically, a vertex v_1 is paired with the edge $\{v_1, v_2\}$ towards its lowest neighbor v_2. (If no neighbor lower than v_1 exists, then v_1 is not paired with an edge.) Furthermore consider the triangles $\{v_1, v_2, v_3\}$ and $\{v_1, v_2, v_3'\}$ incident to an edge $\{v_1, v_2\}$. This edge is paired with the face $\{v_1, v_2, v_{\min}\}$ where $v_{\min}$ is the lowest vertex among v_3 and v_3'. (If none of v_3 and v_3' are lower than both v_1 and v_2, then $\{v_1, v_2\}$ is not paired to a face.) A vertex, edge or face that is not paired with anything is called *critical*; critical vertices, edges and faces are minima, saddles and maxima, respectively (see Fig. 1a–c). The *ascending manifold* of a minimum v is obtained by traversing reversed gradient pairs, starting from v. The *descending manifold* of a maximum v is obtained by traversing gradient pairs, starting from v.

KDS. We aim to construct a KDS to maintain the minima, saddles and maxima, and the ascending and descending manifolds as the vertices continuously change their height. We assume that at no point in time, three vertices have the same height. Our data structure is inspired by the one proposed by Agarwal *et al.* for maintaining contour trees kinetically [1]. Like Agarwal *et al.* we use *link-cut trees*, a data structure that stores a forest of rooted trees dynamically, supporting edge insertions and deletions. Furthermore, the root of each tree can be set and for any vertex the root of its tree can be found. All of these operations take logarithmic time.

To maintain the ascending and descending manifolds, we use two link-cut trees, $T_\downarrow$ and $T_\uparrow$ (see Fig. 1d). $T_\downarrow$ represents the vertex-to-edge gradient pairs. Specifically, $T_\downarrow$ contains a vertex for each vertex in K, and it contains the edge $\{v_1, v_2\}$ for each vertex-to-edge gradient pair $(v_1, \{v_1, v_2\})$. In the static setting discussed by Shivashankar *et al.*, the ascending

* The authors are supported by the Netherlands Organisation for Scientific Research (NWO) under project no. 639.023.208 (T.O., W.S., and B.S.) and no. 639.021.541 (K.V.).

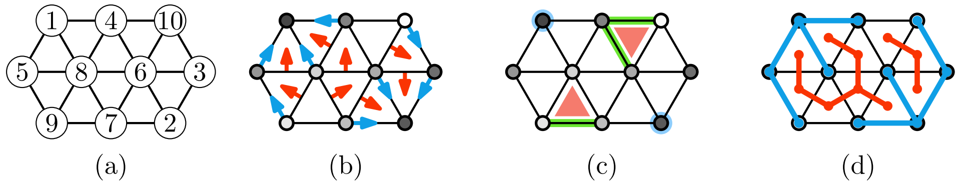

Figure 1 **(a)** A terrain with vertex heights; **(b)** vertex-edge (blue) and edge-face (red) gradient pairs; **(c)** minima (blue), saddles (green) and maxima (red); **(d)** $T_\downarrow$ (blue) and $T_\uparrow$ (red).

manifolds are computed by a BFS starting from each minimum, traversing reversed gradient pairs. Such a BFS corresponds to traversing one complete tree in $T_\downarrow$. Hence, each tree in $T_\downarrow$ represents an ascending manifold; we ensure that its minimum is the root of the tree.

$T_\uparrow$ represents the edge-to-face gradient pairs. Specifically, $T_\uparrow$ contains a vertex for each face in K, and it contains the edge $(\{v_1, v_2, v_3\}, \{v_1, v_2, v_3'\})$ for each edge-to-face gradient pair $(\{v_1, v_2\}, \{v_1, v_2, v\})$. Again, as this mirrors the BFS in the static setting, each tree in $T_\uparrow$ represents a descending manifold; we ensure that its maximum is the root of the tree.

Event handling. We first show how to maintain the set of vertex-edge gradient pairs; that is, $T_\downarrow$. Changes in the vertex-edge gradient pairs happen because the lowest neighbor of a vertex changes. Specifically, when the lowest neighbor of vertex v changes, v needs to be paired with its incident edge that is now lowest. To track this information, we store a tournament tree for each vertex v, to maintain its lowest neighbor. This tournament tree contains v's neighboring vertices and v itself. This leads to three types of events: the lowest neighbor can move from one neighbor v_1 to another neighbor v_2 (in which case we update $T_\downarrow$ by deleting $\{v, v_1\}$ and inserting $\{v, v_2\}$), the lowest neighbor can move from a neighbor v_1 to v itself (in which case we delete $\{v, v_1\}$ from $T_\downarrow$), or the lowest neighbor can move from v to a neighbor v_1 (in which case we insert $\{v, v_1\}$ into $T_\downarrow$). Several such events can happen at the same time, in which case we handle them one by one. To avoid adding cycles to $T_\downarrow$, we first execute all edge deletions, and then all insertions. Similarly we maintain $T_\uparrow$, by maintaining for each edge $\{v_1, v_2\}$ which of v_1, v_2, v_3 and v_3' is the lowest. After an event has been handled, we can locally determine which vertices, edges and faces in the neighborhood are minima, saddles and maxima, respectively, and mark them as such.

Running time. Because an event influences only the neighborhood of a single vertex or face, per event only a constant number of link / cut operations need to be carried out. Assuming the maximum vertex degree in K is bounded by a constant, events can be processed in $O(\log n)$ time each. Hence, if there are k changes to the MS-complex, our KDS can compute those in $O(k \log n)$ time in total.

References

1. P. K. Agarwal, T. Mølhave, M. Revsbæk, I. Safa, Y. Wang, and J. Yang. Maintaining contour trees of dynamic terrains. In *Proc. 31st International Symposium on Computational Geometry*, pages 796–811, 2015.
2. R. Forman. A user's guide to discrete Morse theory. *Séminaire Lotharingien de Combinatoire*, 48:1–35, 2002.
3. N. Shivashankar, Senthilnathan M, and V. Natarajan. Parallel computation of 2D Morse-Smale complexes. *IEEE Transactions on Visualization and Computer Graphics*, 18(10):1757–1770, 2012.