A framework for algorithm stability and its application to kinetic Euclidean MSTs

A Framework for Algorithm Stability and its Application to Kinetic Euclidean MSTs *

Wouter Meulemans, Bettina Speckmann, Kevin Verbeek, and Jules Wulms

Dept. of Mathematics and Computer Science, TU Eindhoven, The Netherlands
[w.meulemans | b.speckmann | k.a.b.verbeek | j.j.h.m.wulms]@tue.nl

1 Introduction

The performance of a particular algorithm is usually judged with respect to a variety of criteria, with the two most common being solution quality and running time. In the context of algorithms for time-varying data, a third important criterion is *stability*. We say that an algorithm is *stable* if small changes in the input result in small changes in the output. The stability of algorithms or methods has been well-studied in a variety of research areas, such as numerical analysis, machine learning, control systems, and topology. In contrast, the stability of combinatorial algorithms for time-varying data has received little attention in the theoretical computer science community so far. Here it is of particular interest to understand the tradeoffs between solution quality, running time, and stability. As an example, consider maintaining a minimum spanning tree of a set of moving points. If the points move, it might have to frequently change significantly. On the other hand, if we start with an MST for the input point set and then never change it combinatorially as the points move, the spanning tree we maintain is very stable – but over time it can devolve to a low quality and very long spanning tree.

Our goal, and the focus of this paper, is to understand the possible tradeoffs between solution quality and stability. This is in contrast to earlier work on stability in other research areas, such as the ones mentioned above, where stability is usually considered in isolation. Since there are currently no suitable tools available to formally analyze tradeoffs involving stability, we introduce a new analysis framework. Our framework allows for three types of stability analysis with increasing degrees of complexity: event stability, topological stability, and Lipschitz stability. We demonstrate the use of our stability framework by applying it to kinetic Euclidean minimum spanning trees. We believe that there are many interesting and relevant questions to be solved in the general area of algorithmic stability analysis and we hope that our framework is a first meaningful step towards tackling them.

Related work. Stability is a natural point of concern in more visual and applied research areas such as graph drawing, (geo-)visualization, and automated cartography. For example, in dynamic map labelling [2], the *consistent dynamic labelling* model allows a label to appear and disappear only once, making it very stable. There are very few theoretical results, with the noteworthy exception of so-called simultaneous embeddings [3] in graph drawing, which can be seen as a very restricted model of stability. However, none of these results offer any real structural insight into the tradeoff between solution quality and stability.

In computational geometry there are a few results on the tradeoff between solution quality and stability. Specifically, Durocher and Kirkpatrick [5] study the stability of centers of kinetic point sets, and define the notion of κ-stable center functions, which is closely related

* W. Meulemans and J. Wulms are (partially) supported by the Netherlands eScience Center (NLeSC) under grant number 027.015.G02. B. Speckmann and K. Verbeek are supported by the Netherlands Organisation for Scientific Research (NWO) under project no. 639.023.208 and no. 639.021.541, resp.

34th European Workshop on Computational Geometry, Berlin, Germany, March 21–23, 2018.
This is an extended abstract of a presentation given at EuroCG'18. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

to our concept of Lipschitz stability. In later work [6] they consider the tradeoff between the solution quality of Euclidean 2-centers and a bound on the velocity with which they can move. De Berg *et al.* [4] show similar results in the black-box KDS model. One can argue that the KDS framework [8] already indirectly considers stability in a limited form, namely as the number of *external events*. However, the goal of a KDS is typically to reduce the running time of the algorithm, and rarely to sacrifice the running time or solution quality to reduce the number of external events.

2 Stability framework

Intuitively, we can say that an algorithm is stable if small changes in the input lead to small changes in the output. More formally, let Π be an optimization problem that, given an input instance I from a set $\mathcal{I}$, asks for a feasible solution S from a set $\mathcal{S}$ that minimizes (or maximizes) some optimization function $f\colon \mathcal{I} \times \mathcal{S} \to \mathbb{R}$. An algorithm $\mathcal{A}$ for Π can be seen as a function $\mathcal{A}\colon \mathcal{I} \to \mathcal{S}$. Similarly, the optimal solutions for Π can be described by a function $\mathrm{OPT}\colon \mathcal{I} \to \mathcal{S}$. To define the stability of an algorithm, we need to quantify changes in the input instances and in the solutions. We can do so by imposing a metric on $\mathcal{I}$ and $\mathcal{S}$. Let $d_{\mathcal{I}}\colon \mathcal{I} \times \mathcal{I} \to \mathbb{R}_{\geq 0}$ be a metric for $\mathcal{I}$ and let $d_{\mathcal{S}}\colon \mathcal{S} \times \mathcal{S} \to \mathbb{R}_{\geq 0}$ be a metric for $\mathcal{S}$. We can then define the *stability* of an algorithm $\mathcal{A}\colon \mathcal{I} \to \mathcal{S}$ as follows.

$$\mathrm{St}(\mathcal{A}) = \max_{I,I' \in \mathcal{I}} \frac{d_{\mathcal{S}}(\mathcal{A}(I), \mathcal{A}(I'))}{d_{\mathcal{I}}(I, I')} \tag{1}$$

This definition for stability is closely related to that of the multiplicative distortion of metric embeddings, where $\mathcal{A}$ induces a metric embedding from the metric space $(\mathcal{I}, d_{\mathcal{I}})$ into $(\mathcal{S}, d_{\mathcal{S}})$. The lower the value for $\mathrm{St}(\mathcal{A})$, the more stable we consider the algorithm $\mathcal{A}$ to be. There are many other ways to define the stability of an algorithm given the metrics, but the above definition suffices for our purpose.

For many optimization problems, the function OPT may be very unstable. This suggests an interesting tradeoff between the stability of an algorithm and the solution quality. Unfortunately, the generic formulation of stability provided above is very unwieldy. It is not always clear how to define metrics $d_{\mathcal{I}}$ and $d_{\mathcal{S}}$ such that meaningful results can be derived. Additionally, it is not obvious how to deal with optimization problems with continuous input and discrete solutions, where the algorithm is inherently discontinuous, and thus the stability is unbounded by definition. Finally, analyses of this form are often very complex, and it is not straightforward to formulate a simplified version of the problem. In our framework we hence distinguish three types of stability analysis: event stability, topological stability, and Lipschitz stability.

Event stability follows the setting of kinetic data structures (KDS). That is, the input (a set of moving objects) changes continuously as a function over time. However, contrary to typical KDSs where a constraint is imposed on the solution quality, we aim to enforce the stability of the algorithm. For event stability we simply disallow the algorithm to change the solution too rapidly. Doing so directly is problematic, but we formalize this approach using the concept of k-optimal solutions. As a result, we can obtain a tradeoff between stability and quality that can be tuned by the parameter k. Note that event stability captures only *how often* the solution changes, but not *how much* the solution changes at each event.

Topological stability takes a first step towards the generic setup described above. However, instead of measuring the amount of change in the solution using a metric, we merely require the solution to behave continuously. To do so we only need to define a topology on the solution

space $\mathcal{S}$ that captures stable behavior. Surprisingly, even though we ignore the amount of change in a single time step, this type of analysis still provides meaningful information on the tradeoff between solution quality and stability. In fact, the resulting tradeoff can be seen as a lower bound for any analysis involving metrics that follow the used topology.

Lipschitz stability finally captures the generic setup described above. As the name suggests, we require the algorithm to be Lipschitz continuous and we provide an upper bound on the Lipschitz constant, which is equivalent to $\mathrm{St}(\mathcal{A})$. We are again interested in the quality of the solutions that can be obtained with any Lipschitz stable algorithm. Given the complexity of this type of analysis, a complete tradeoff for any value of the Lipschitz constant is typically out of reach, but results for sufficiently small or large values can be of interest.

Remark. Our framework makes the assumption that an algorithm is a function $\mathcal{A}\colon \mathcal{I} \to \mathcal{S}$. However, in a kinetic setting this is not necessarily true, since the algorithm has *history*. More precisely, for some input instance I, a kinetic algorithm may produce different solutions for I based on the instances processed earlier. We generally allow this behavior, and for event stability this behavior is even crucial. However, for the sake of simplicity, we will treat an algorithm as a function. We also generally assume in our analysis that the input is time-varying, that is, the input is a function over time, or follows a trajectory through the input space $\mathcal{I}$. Again, for the sake of simplicity, this is not always directly reflected in our definitions. Beyond that, we operate in the black-box model, in the sense that the algorithm does not know anything about future instances.

In the remainder we focus on topological stability, all omitted material (the description of event and Lipschitz stability, as well as proofs) can be found in the full version [9].

3 Topological stability

Topological stability analysis is applicable to a wide variety of problems and enforces continuous changes to the solution. Even though it does not capture stability in its entirety, as changes can happen in infinitesimally short time, topological stability still illustrates clearly how solutions and their quality have to change as the input changes.

3.1 Topological stability analysis

Let Π be an optimization problem with input instances $\mathcal{I}$, solutions $\mathcal{S}$, and optimization function f. An algorithm $\mathcal{A}\colon \mathcal{I} \to \mathcal{S}$ is *topologically stable* if, for any (continuous) path $\pi\colon [0,1] \to \mathcal{I}$ in $\mathcal{I}$, $\mathcal{A}\pi$ is a (continuous) path in $\mathcal{S}$. To properly define a (continuous) path in $\mathcal{I}$ and $\mathcal{S}$ we need to specify a topology $\mathcal{T}_\mathcal{I}$ on $\mathcal{I}$ and a topology $\mathcal{T}_\mathcal{S}$ on $\mathcal{S}$. Alternatively we could specify metrics $d_\mathcal{I}$ and $d_\mathcal{S}$, but this is typically more involved. We then want to analyze the approximation ratio of any topologically stable algorithm with respect to OPT. That is, we are interested in the ratio

$$\rho_{\mathrm{TS}}(\Pi, \mathcal{T}_\mathcal{I}, \mathcal{T}_\mathcal{S}) = \inf_{\mathcal{A}} \sup_{I \in \mathcal{I}} \frac{f(I, \mathcal{A}(I))}{f(I, \mathrm{OPT}(I))} \tag{2}$$

where the infimum is taken over all topologically stable algorithms. Naturally, if OPT is already topologically stable, then this type of analysis does not provide any insight and the ratio is simply 1. However, in many cases, OPT is not topologically stable. The above analysis can also be applied if the solution space (or the input space) is discrete. In such cases, continuity can often be defined using the graph topology of so-called flip graphs, for example,

based on edge flips for triangulations or rotations in rooted binary trees. We can represent a graph as a topological space by representing vertices by points, and representing every edge of the graph by a copy of the unit interval $[0, 1]$. These intervals are glued together at the vertices. In other words, we consider the corresponding simplicial 1-complex. Although the points in the interior of the edges of this topological space do not necessarily represent proper solutions, we can still use this topological space in Equation 2 by extending f over the edges via linear interpolation. It is not hard to see that we need to consider only the vertices of the flip graph (which do represent proper solutions) to compute the topological stability ratio.

Lower bounds. When proving lower bounds on the topological stability of a problem we want to force any algorithm that continuously updates the solution to produce a particularly bad intermediate result. We first show that updating a certain configuration continuously will always result in an intermediate solution of low quality, no matter which algorithm is used. We then also provide a particular motion that forces an update to the solution in that configuration. Updating the solution at any other point during the motion should lead to an even worse result. The motion and the described configuration together allow us to prove a lower bound on the topological stability ratio. In this abstract we do not describe the motions for the lower bound proofs; the complete proofs can be found in the full version [9].

3.2 Topological stability of EMSTs

Our input consists of a set of n points where each point has a trajectory. We require that the trajectories are continuous. The goal is to maintain a combinatorial description of a short spanning tree on these points, whose length stays close to optimal. To define this properly, we need to define a topology on the input space, but for a kinetic point set with n points in d dimensions we can simply use the standard topology on $\mathbb{R}^{dn}$ as $\mathcal{T}_\mathcal{I}$. To apply topological stability analysis, we also need to specify a topology on the (discrete) solution space. As the points move, the minimum spanning tree may have to change at some point in time by removing one edge and inserting another edge. Since these two edges may be very far apart, we do not consider this operation to be stable or continuous. Instead we specify the topology of $\mathcal{S}$ using a flip graph, where the operations are either *edge slides* or *edge rotations* [1, 7]. The optimization function f, measuring the quality of the EMST, is naturally defined for the vertices of the flip graph as the length of the spanning tree, and we use linear interpolation to define f on the edges of the flip graph. For edge slides and rotations we provide upper and lower bounds on $\rho_{\mathrm{TS}}(\mathrm{EMST}, \mathcal{T}_\mathcal{I}, \mathcal{T}_\mathcal{S})$.

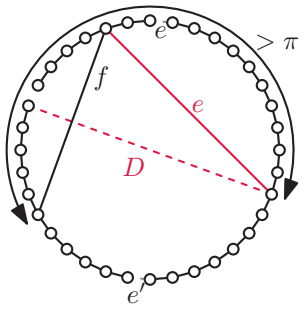
Edge slides. An edge slide is defined as the operation of moving one endpoint of an edge to one of its neighboring vertices along the edge to that neighbor. More formally, an edge (u, v) can be replaced by (u, w) if w is a neighbor of v and $w \neq u$. Since this operation is very local, we consider it to be stable. Note that after every edge slide the tree must still be connected.

▶ **Lemma 1.** *If $\mathcal{T}_\mathcal{S}$ is defined by edge slides, then $\rho_{\mathrm{TS}}(EMST, \mathcal{T}_\mathcal{I}, \mathcal{T}_\mathcal{S}) \leq \frac{3}{2}$.*

Proof. Consider a time where the EMST has to be updated by removing an edge e and inserting an edge e', where $|e| = |e'|$. Note that e and e' form a cycle C with other edges of the EMST. We now slide edge e to edge e' by sliding it along the vertices of C. Let x be the longest intermediate edge when sliding from e to e' (see Fig. 1(a)). To allow x to be as long as possible with respect to the length of the EMST and as such achieving an upper bound on ρ_{TS}, the EMST should be fully contained in C. By the triangle inequality we get that $2|x| \leq |C|$. Since the length of the EMST is $\mathrm{OPT} = |C| - |e|$, we get that $|x| \leq \mathrm{OPT}/2 + |e|/2$. Thus, the length of the intermediate tree is $|C| - 2|e| + |x| = \mathrm{OPT} - |e| + |x| \leq \frac{3}{2}\,\mathrm{OPT}$. ◀

▶ **Lemma 2.** *If $\mathcal{T}_S$ is defined by edge slides, then $\rho_{\mathrm{TS}}(EMST, \mathcal{T}_I, \mathcal{T}_S) \geq \frac{\pi+1}{\pi}$.*

Proof. Consider a point in time where the EMST has to be updated by removing an edge e and inserting an edge e', where $|e|$ is very small. Let the remaining points be arranged in a circle with diameter D, as shown in the figure on the right. We get that $OPT < \pi D$, where OPT is the length of the EMST. Simply sliding e to e' will always grow e to be nearly the diameter of the circle at some point, as shown by the red dashed line. More precisely, e will grow to length at least $D - \varepsilon$, and we can make ε arbitrarily small by using a sufficient number of points. Alternatively, e (in the red configuration) can take a shortcut by sliding over another edge f. This is only beneficial if $|e| + |f| < D - \varepsilon$. However, if f helps e to avoid becoming a diameter of the circle, then e and f, as chords, must span an angle larger than π together. Hence, by triangle inequality, $|e| + |f| \geq D$. Thus, for any $\varepsilon > 0$, $\rho_{\mathrm{TS}}(\mathrm{EMST}, \mathcal{T}_I, \mathcal{T}_S) \geq \frac{OPT+D}{OPT} - \varepsilon > \frac{OPT+OPT/\pi}{OPT} - \varepsilon = \frac{\pi+1}{\pi} - \varepsilon \approx 1.318 - \varepsilon$. ◀



Edge rotations. Edge rotations are a generalization of edge slides, that allow one endpoint of an edge to move to any other vertex. These operations are clearly not as stable as edge slides, but they are still more stable than the deletion and insertion of arbitrary edges.

▶ **Lemma 3.** *If $\mathcal{T}_S$ is defined by edge rotations, then $\rho_{\mathrm{TS}}(EMST, \mathcal{T}_I, \mathcal{T}_S) \leq \frac{4}{3}$.*

Proof. Consider a time where the EMST has to be updated by removing an edge $e = (u, v)$ and inserting an edge $e' = (u', v')$, where $|e| = |e'|$. Note that e and e' form a cycle C with other edges of the EMST. We now rotate edge e to edge e' along some of the vertices of C. Let x be the longest intermediate edge when rotating from e to e'. To allow x to be as long as possible with respect to the length of the EMST, the EMST should be fully contained in C. We argue that $|x| \leq \mathrm{OPT}/3 + |e|$, where OPT is the length of the EMST. Removing e and e' from C splits C into two parts, where we assume that u and u' (v and v') are in the left (right) part. First assume that one of the two parts has length at most $\mathrm{OPT}/3$. Then we can rotate e to (u, v'), and then to e', which implies that $|x| = |(u, v')| \leq \mathrm{OPT}/3 + |e|$ by the triangle inequality (see Fig. 1(b)). Now assume that both parts have length at least $\mathrm{OPT}/3$. Let $e_L = (a, b)$ be the edge in the left part that contains the midpoint of that part, and let $e_R = (c, d)$ be the edge in the right part that contains the midpoint of that part, where u_L and u_R are closest to e (see Fig. 1(c)). Furthermore, let Z be the length of $C \setminus \{e, e', e_L, e_R\}$.

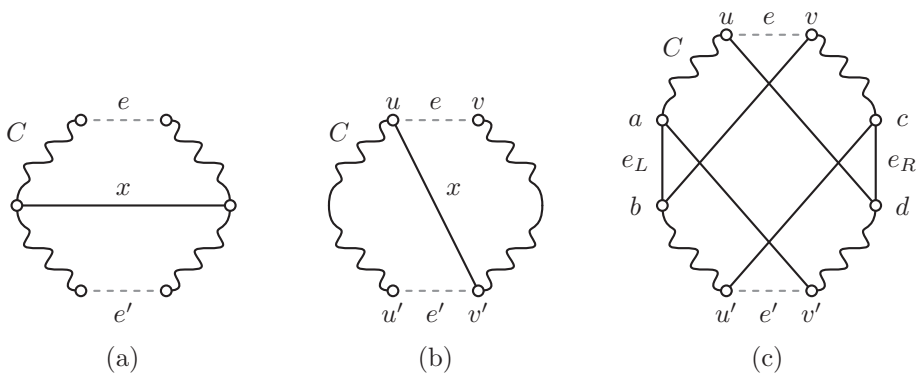
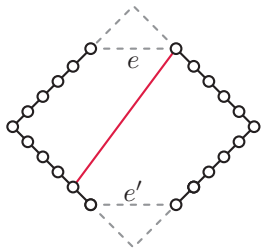


Figure 1 (a): Illustration for Lemma 1. (b) and (c): Illustrating the two cases for Lemma 3.

Now consider the potential edges (u, d), (v, b), (u', c), and (v', a). By the triangle inequality, the sum of the lengths of these edges is at most $4|e| + 2|e_L| + 2|e_R| + Z$. Thus, one of these potential edges has length at most $|e| + |e_L|/2 + |e_R|/2 + Z/4$. Without loss of generality let (u, d) be that edge (the construction is fully symmetric). We can now rotate e to (u, d), then to (u', d), and finally to e'. As each part of C has length at most $2\,\mathrm{OPT}/3$, we get that $|(u', d)| \leq \mathrm{OPT}/3 + |e|$ by construction. Furthermore we have that $\mathrm{OPT} = |e| + |e_L| + |e_R| + Z$. Thus, $|(u, d)| \leq |e| + |e_L|/2 + |e_R|/2 + Z/4 = \mathrm{OPT}/3 + 2|e|/3 + |e_L|/6 + |e_R|/6 - Z/12$. Since e needs to be removed to update the EMST, it must be the longest edge in C. Therefore $|(u, d)| \leq \mathrm{OPT}/3 + |e|$, which shows that $|x| \leq \mathrm{OPT}/3 + |e|$. Since the length of the intermediate tree is $\mathrm{OPT} - |e| + |x| \leq \frac{4}{3}\,\mathrm{OPT}$, we obtain that $\rho_{\mathrm{TS}}(\mathrm{EMST}, \mathcal{T}_{\mathcal{I}}, \mathcal{T}_{\mathcal{S}}) \leq \frac{4}{3}$. ◀

▶ **Lemma 4.** *If $\mathcal{T}_{\mathcal{S}}$ is defined by edge rotations, then, $\rho_{\mathrm{TS}}(EMST, \mathcal{T}_{\mathcal{I}}, \mathcal{T}_{\mathcal{S}}) \geq \frac{10-2\sqrt{2}}{9-2\sqrt{2}}$.*

Proof. Consider a point in time where the EMST has to be updated by removing an edge e and inserting an edge e'. Let the remaining points be arranged in a diamond shape as shown in the figure on the right, where the side length of the diamond is 2, and $|e| = |e'| = 1$. Now we define a *top-connector* as an edge that intersects the vertical diagonal of the diamond, but is completely above the horizontal diagonal of the diamond. A *bottom-connector* is defined analogously, but must be completely below the horizontal diagonal. Finally, a *cross-connector* is an edge that hits both diagonals of the diamond. Note that a cross-connector has length at least 2, and a top- or bottom-connector has length at least $|e| = 1$. In the considered update, we start with a top-connector and end with a bottom-connector. Since we cannot rotate from a top-connector to a bottom-connector in one step, we must reach a state that either has both a top-connector and a bottom-connector, or a single cross-connector. In both options the length of the spanning tree is $10 - 2\sqrt{2}$, while the minimum spanning tree has length $9 - 2\sqrt{2}$. Thus $\rho_{\mathrm{TS}}(\mathrm{EMST}, \mathcal{T}_{\mathcal{I}}, \mathcal{T}_{\mathcal{S}}) \geq \frac{10-2\sqrt{2}}{9-2\sqrt{2}} \approx 1.162$. ◀



References

1 O. Aichholzer, F. Aurenhammer, and F. Hurtado. Sequences of spanning trees and a fixed tree theorem. *Comp. Geom. Theory Appl.*, 21(1-2):3–20, 2002.

2 K. Been, M. Nöllenburg, S.-H. Poon, and A. Wolff. Optimizing active ranges for consistent dynamic map labeling. *Comp. Geom. Theory Appl.*, 43(3):312–328, 2010.

3 P. Brass, E. Cenek, C. A. Duncan, A. Efrat, C. Erten, D. P. Ismailescu, S. G. Kobourov, A. Lubiw, and J. S. Mitchell. On simultaneous planar graph embeddings. *Comp. Geom. Theory Appl.*, 36(2):117–130, 2007.

4 M. de Berg, M. Roeloffzen, and B. Speckmann. Kinetic 2-centers in the black-box model. In *Proc. 29th Symp. Comp. Geom.*, pages 145–154, 2013.

5 S. Durocher and D. Kirkpatrick. The Steiner centre of a set of points: Stability, eccentricity, and applications to mobile facility location. *I. J. Comp. Geom. Appl.*, 16(04):345–371, 2006.

6 S. Durocher and D. Kirkpatrick. Bounded-velocity approximation of mobile Euclidean 2-centres. *I. J. Comp. Geom. Appl.*, 18(03):161–183, 2008.

7 W. Goddard and H. C. Swart. Distances between graphs under edge operations. *Discrete Mathematics*, 161(1-3):121–132, 1996.

8 L. J. Guibas. Kinetic data structures. In D. P. Mehta and S. Sahni (eds), *Handbook of Data Structures and Applications*, pages 23.1–18. Chapman and Hall/CRC, 2004.

9 W. Meulemans, B. Speckmann, K. Verbeek, and J. Wulms. A framework for algorithm stability. *CoRR*, abs/1704.08000, 2017. URL: `http://arxiv.org/abs/1704.08000`.