# 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
[w.meulemans|b.speckmann|k.a.b.verbeek|j.j.h.m.wulms]@tue.nl

**Abstract.** We say that an algorithm is *stable* if small changes in the input result in small changes in the output. This kind of algorithm stability is particularly relevant when analyzing and visualizing time-varying data. Stability in general plays an important role in a wide variety of areas, such as numerical analysis, machine learning, and topology, but is poorly understood in the context of (combinatorial) algorithms.
In this paper we present a framework for analyzing the stability of algorithms. We focus in particular on the tradeoff between the stability of an algorithm and the quality of the solution it computes. 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.

## 1 Introduction

With recent advances in sensing technology, vast amounts of *time-varying data* are generated, processed, and analyzed on a daily basis. Hence there is a great need for algorithms that can operate efficiently on time-varying data and that can offer guarantees on the quality of analysis results. A specific relevant subset of time-varying data consists of so-called *motion data*: geolocated, and hence geometric, time-varying data. To deal with the challenges of motion data, Basch *et al.* [3] in 1999 introduced the *kinetic data structures* (KDS) framework. Kinetic data structures efficiently maintain a (combinatorial) structure on a set of moving objects. The KDS framework has sparked a significant amount of research, resulting in many efficient algorithms for motion data.

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*. Whenever analysis results need to be communicated to humans, for example via visual representations, it is important that these results

* 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, respectively.

are *stable*: small changes in the data result in small changes in the output. These changes in the output are continuous or discrete depending on the algorithm: graphs usually undergo discrete changes while a convex hull of moving points changes continuously. Sudden changes in the visual representation of data disrupt the so-called *mental map* [16] of the user and prevent the recognition of temporal patterns. Stability also plays a role if changing the result is costly in practice (e.g. in physical network design), where frequent significant changes to the network are prohibitively expensive.

The stability of algorithms or methods has been well-studied in a variety of research areas, such as numerical analysis [14], machine learning [5], control systems [2], and topology [7]. 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. 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.

**Results and organization.** We present a framework to analyze the stability of (combinatorial) algorithms. As a first step, we limit ourselves to analyzing the tradeoff between stability and solution quality, omitting running time from consideration. Our framework allows for three types of stability analysis of increasing degrees of complexity: *event stability*, *topological stability*, and *Lipschitz stability*. It can be applied both to motion data and to more general time-varying data. We demonstrate the use of our stability framework by applying it to the problem of kinetic Euclidean minimum spanning trees (EMSTs). Some of our results for kinetic EMSTs are directly more widely applicable.

In Section 2 we give an overview of our framework for algorithm stability. In Sections 3, 4, and 5 we describe event stability, topological stability, and Lipschitz stability, respectively. In each of these sections we first describe the respective type of stability analysis in a generic setting, followed by specific results using that type of stability analysis on the kinetic EMST problem. In Section 6 we make some concluding remarks on our stability framework. Omitted proofs can be found in the full version of the paper.

**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 [4], 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 [6] 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 [9] study the stability of centers of kinetic point sets, and define the notion of $\kappa$-stable center functions, which is closely related to our concept of Lipschitz stability. In later work [10] 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.* [8] show similar results in the black-box KDS model. One can argue that the KDS framework [13] 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.

Kinetic Euclidean minimum spanning trees have been studied extensively. Katoh *et al.* [15] proved an upper bound of $O(n^3 2^{\alpha(n)})$ for the number of external events of EMSTs of $n$ linearly moving points, where $\alpha(n)$ is the inverse Ackermann function. The best known lower bound for external events of EMSTs in $d$ dimensions is $\Omega(n^d)$ [18]. There also exists a lot of related work on approximations of EMSTs and related structures like Delaunay triangulations, but the number of external events still remains at least roughly $\Omega(n^2)$. Our stability framework allows us to reduce the number of external events even further and to still state something meaningful about the quality of the resulting EMSTs.

## 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 $\Pi$ 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 $\Pi$ can be seen as a function $\mathcal{A}\colon \mathcal{I} \to \mathcal{S}$. Similarly, the optimal solutions for $\Pi$ 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 then 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.

While these are the conditions under which we use our framework, it can be applied in a variety of algorithmic settings, such as streaming algorithms and algorithms with dynamic input.

## 3 Event stability

The simplest form of stability is event stability. Like the number of external events in KDSs, event stability captures only how often the solution changes.

### 3.1 Event stability analysis

Let $\Pi$ be an optimization problem with a set of input instances $\mathcal{I}$, a set of solutions $\mathcal{S}$, and optimization function $f\colon \mathcal{I} \times \mathcal{S} \to \mathbb{R}$. Following the framework of kinetic data structures, we assume that the input instances include certain parameters that can change as a function of time. To apply the event stability analysis, we require that all solutions have a combinatorial description, that is, the solution description does not use the time-varying parameters of the input instance. We further require that every solution $S \in \mathcal{S}$ is feasible for every input instance $I \in \mathcal{I}$. This automatically disallows any insertions or deletions of elements. Note that an insertion or a deletion would typically force an event, and thus including this aspect in our stability analysis does not seem useful.

For example, in the setting of kinetic EMSTs, the input instances would consist of a fixed set of points. The coordinates of these points can then change as a function over time. A solution of the kinetic EMST problem consists of the combinatorial description of a tree graph on the set of input points. Note that every tree graph describes a feasible solution for any input instance, if we do not insist on any additional restrictions like, e.g., planarity. The minimization function $f$ then simply measures the total length of the tree, for which we do need to use the time-varying parameters of the problem instance.

Rather than directly restricting the quality of the solutions, we aim to restrict the stability of any algorithm. To that end, we introduce the concept of $k$-optimal solutions. Let $d_{\mathcal{I}}$ be a metric on the input instances, and let $\mathrm{OPT}\colon \mathcal{I} \to \mathcal{S}$ describe the optimal solutions. We say that a solution $S \in \mathcal{S}$ is *$k$-optimal* for an instance $I \in \mathcal{I}$ if there exists an input instance $I' \in \mathcal{I}$ such that $f(I', S) = f(I', OPT(I'))$ and $d_{\mathcal{I}}(I, I') \leq k$. With this definition any optimal solution is always 0-optimal. Note that this definition requires a form of normalization on the metric $d_{\mathcal{I}}$, similar to that of e.g. smoothed analysis [19]. We therefore require that there exists a constant $c$ such that every solution $S \in \mathcal{S}$ is $c$-optimal for every instance $I \in \mathcal{I}$. For technical reasons we require the latter condition to hold only for some time interval $[0, T]$ of interest. Note that the concept of $k$-optimal solutions is closely related to *backward error analysis* in numerical analysis.

Following the framework of kinetic data structures, we typically require the functions of the time-varying parameters to be well-behaved (e.g., polynomial functions), for otherwise we cannot derive meaningful bounds. The event stability analysis then considers two aspects. First, we analyze how often the solution needs to change to maintain a $k$-optimal solution for every point in time. Second, we analyze how well a $k$-optimal solution approximates an optimal solution. Typically we are not able to directly obtain good bounds on the approximation ratio, but given certain reasonable assumptions, good approximation bounds as a function of $k$ can be provided.

### 3.2 Event stability for EMSTs

Our input consists of a set of points $P = \{p_1, \ldots, p_n\}$ where each point $p_i$ has a trajectory described by the function $x_i \colon [0, T] \to \mathbb{R}^d$. The goal is to maintain a combinatorial description of a short spanning tree on $P$ that does not change often. We assume that the functions $x_i$ are polynomials with bounded degree $s$.

To use the concept of $k$-optimal solutions, we first need to normalize the coordinates. We simply assume that $x_i(t) \in [0, 1]^d$ for $t \in [0, T]$. This assumption may seem overly restrictive for kinetic point sets, but note that we are only interested in relative positions, and thus the frame of reference may move with the points. Next, we define the metric $d_{\mathcal{I}}$ along the trajectory as follows.

$$d_{\mathcal{I}}(t, t') = \max_i \|x_i(t) - x_i(t')\| \tag{2}$$

Note that this metric, and the resulting definition of $k$-optimal solutions, is not specific to EMSTs and can be used in general for problems with kinetic point sets as input. In our case $\|a - b\|$ denotes the distance between $a$ and $b$ in the (Euclidean) $\ell_2$ norm. Now let $OPT(t)$ be the EMST at time $t$. Then, by definition, $OPT(t)$ is $k$-optimal at time $t'$ if $d_{\mathcal{I}}(t, t') \leq k$. Our approach is now very simple: we compute the EMST and keep that solution as long as it is $k$-optimal, after which we compute the new EMST, and so forth. Below we analyze this approach.

**Number of events.** To bound the number of events, we first need to bound the speed of any point with a polynomial trajectory and bounded coordinates. For this we can use a classic result known as the *Markov Brothers' inequality*.

**Lemma 1 ([17]).** *Let $h(t)$ be a polynomial with degree at most $s$ such that $h(t) \in [0, 1]$ for $t \in [0, T]$, then $|h'(t)| \leq s^2/T$ for all $t \in [0, T]$.*

**Lemma 2.** *For a kinetic point set $P$ with degree-$s$ polynomial trajectories $x_i(t) \in [0, 1]^d$ ($t \in [0, T]$) we need only $O(\frac{s^2}{k})$ changes to maintain a $k$-optimal solution for constant $d$.*

*Proof.* By Lemma 1 the velocity of any point is at most $s^2/T$ in one dimension, and thus at most $\sqrt{d}\, s^2/T = O(s^2/T)$ in $d$ dimensions, assuming $d$ is constant. Now assume that we have computed an optimal solution $S$ for some time $t$. The solution $S$ remains $k$-optimal until one of the points has moved at least $k$ units.

Since the velocity of the points is bounded, this takes at least $\Delta t = kT/s^2$ time, at which point we can recompute the optimal solution. Since the total time interval is of length $T$, this can happen at most $T/\Delta t = s^2/k$ times. □

**Approximation factor.** We cannot expect $k$-optimal solutions to be a good approximation of an optimal EMST's length in general: if all points are within distance $k$ from each other, then all solutions are $k$-optimal. We therefore need to make the assumption that the points are spread out reasonably throughout the motion. To quantify this, we use a measure inspired by the *order-l spread*, as defined in [11]. Let $\text{MINDIST}_l(P)$ be the smallest distance in $P$ between a point and its $l$-th nearest neighbor. We assume that $\text{MINDIST}_l(P) \geq 1/\Delta_l$ throughout the motion, for some value of $\Delta_l$. We can use this assumption to give a lower bound on the length of the EMST. Pick an arbitrary point and remove all points from $P$ that are within distance $1/\Delta_l$, and repeat this process until the smallest distance is at least $1/\Delta_l$. By our assumption, we remove at most $l-1$ points for each chosen point, so we are left with at least $n/l$ points. The length of the EMST on P is at least the length of the EMST on the remaining $n/l$ points, which has length $\Omega(\frac{n}{l\Delta_l})$.

**Lemma 3.** *A $k$-optimal solution of the EMST problem on a set of $n$ points $P$ is an $O(1+kl\Delta_l)$-approximation of the EMST, under the assumption that $\text{MINDIST}_l(P) \geq 1/\Delta_l$.*

*Proof.* Let $S$ be a $k$-optimal solution of $P$ and let OPT be an optimal solution of $P$. By definition there is a point set $P'$ for which the length of solution $S$ is at most that of OPT. Since $d_{\mathcal{I}}(P, P') \leq k$, the length of each edge can grow or shrink by at most $2k$ when moving from $P'$ to $P$. Therefore we can state that $f(P,S) \leq f(P,\text{OPT}) + 4kn$. Now, using the lower bound on the length of an EMST, we obtain the following.

$$\begin{aligned} f(P,\text{OPT}) + 4kn &\leq f(P,\text{OPT}) + 4kO(f(P,\text{OPT})l\Delta_l) \\ &= O(1+kl\Delta_l) \cdot f(P,\text{OPT}) \end{aligned}$$ □

Note that there is a clear tradeoff between the approximation ratio and how restrictive the assumption on the spread is. Regardless, we can obtain a decent approximation while only processing a small number of events. If we choose reasonable values $k = O(1/n)$, $l = O(1)$, and $\Delta_l = O(n)$, then our results show that, under the assumptions, a constant-factor approximation of the EMST can be maintained while processing only $O(n)$ events.

## 4 Topological stability

The event stability analysis has two major drawbacks: (1) it is only applicable to problems for which the solutions are always feasible and described combinatorially, and (2) it does not distinguish between small and large structural changes. Topological stability analysis is applicable to a wide variety of problems and enforces continuous changes to the solution.

### 4.1 Topological stability analysis

Let $\Pi$ 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

$$\rho_{\mathrm{TS}}(\Pi, \mathcal{T}_\mathcal{I}, \mathcal{T}_\mathcal{S}) = \inf_{\mathcal{A}} \sup_{I \in \mathcal{I}} \frac{f(I, \mathcal{A}(I))}{f(I, \mathrm{OPT}(I))} \tag{3}$$

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 represent proper spanning trees, we can still use this topological space in Equation 3 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 represent proper spanning trees) to compute the topological stability ratio.

### 4.2 Topological stability of EMSTs

We use the same setting of the kinetic EMST problem as in Section 3.2, except that we do not restrict the trajectories of the points and we do not normalize the coordinates. We merely require that the trajectories are continuous. 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, 12]. 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 $(u, v)$ in the tree 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 4.** *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. 2(a)). To allow $x$ to be as long as possible with respect to the length of the EMST, the EMST should be fully contained in $C$. 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 5.** *If $\mathcal{T}_{\mathcal{S}}$ is defined by edge slides, then $\rho_{\mathrm{TS}}(EMST, \mathcal{T}_{\mathcal{I}}, \mathcal{T}_{\mathcal{S}}) \geq \frac{\pi+1}{\pi}$.*

*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, as shown in Figure 1(a), such that the farthest distance between any two points is $\mathrm{OPT}/\pi - \varepsilon$, where OPT is the length of the EMST. We can make this construction for any $\varepsilon > 0$ by using enough points and making $e$ and $e'$ arbitrarily short. Simply sliding $e$ to $e'$ will always grow $e$ to be

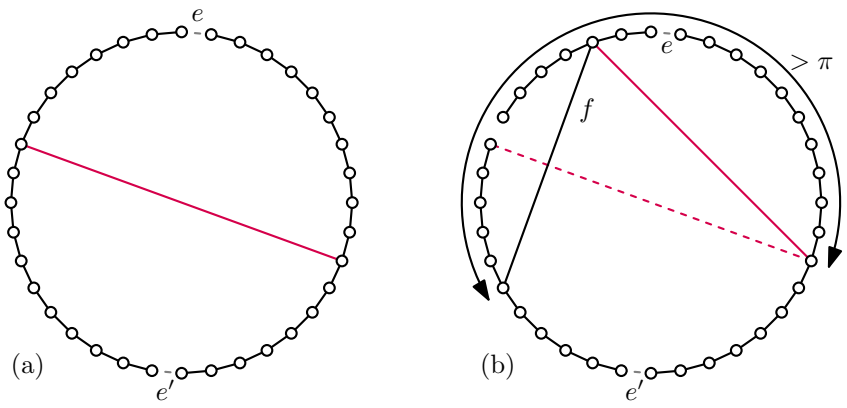



**Fig. 1.** This configuration is a $(\frac{\pi+1}{\pi} - \varepsilon)$-approximation of the EMST. (a) While $e$ slides to $e'$ it becomes the diameter of the circle. (b) Stretching an edge $f$ to form a chord creates an even longer spanning tree.

the diameter of the circle at some point. Alternatively, $e$ can take a shortcut by sliding over another edge $f$ as a chord (see Figure 1(b)). This is only beneficial if $|e| + |f| < \mathrm{OPT}/\pi - \varepsilon$. 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 $\pi$ together. As a result, $|e| + |f| \geq \mathrm{OPT}/\pi - \varepsilon$ by triangle inequality.

A motion of the points that forces $e$ to slide to $e'$ in this particular configuration looks as follows. The points start at $e$ and move at constant speed along the circle, half of the points clockwise and the other half counter clockwise. The speeds are assigned in such a way that at some point all points are evenly spread along the circle. Once all points are evenly spread, they start moving towards $e'$, again along the circle. It is easy to see that, using the arguments above, any additional edges inside the circle must have total length of at least the diameter of the circle at some point throughout the motion. On the other hand, OPT is largest when the points are evenly spread along the circle. Thus, for any $\varepsilon > 0$, $\rho_{\mathrm{TS}}(\mathrm{EMST}, \mathcal{T}_{\mathcal{I}}, \mathcal{T}_{\mathcal{S}}) \geq \frac{\pi+1}{\pi} - \varepsilon \approx 1.318 - \varepsilon$. □

**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 6.** *If $\mathcal{T}_{\mathcal{S}}$ is defined by edge rotations, then $\rho_{\mathrm{TS}}(EMST, \mathcal{T}_{\mathcal{I}}, \mathcal{T}_{\mathcal{S}}) \leq \frac{4}{3}$.*

*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 $u$

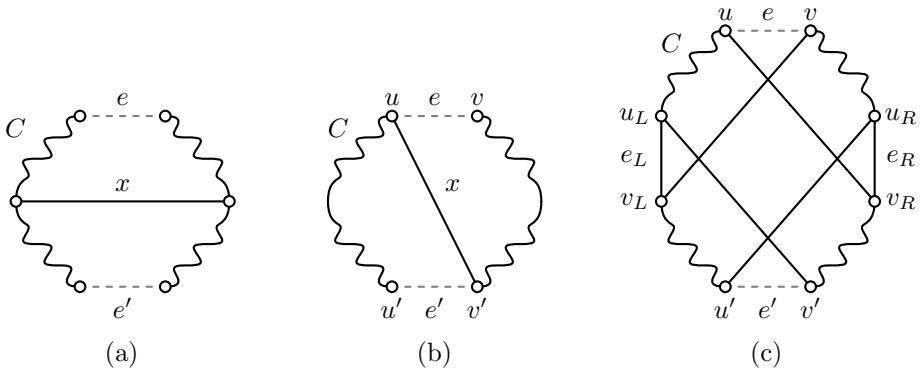
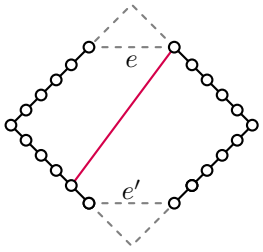



**Fig. 2.** (a): Illustration for Lemma 4. (b) and (c): Illustrating the two cases for Lemma 6.

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. 2(b)). Now assume that both parts have length at least $\mathrm{OPT}/3$. Let $e_L = (u_L, v_L)$ be the edge in the left part that contains the midpoint of that part, and let $e_R = (u_R, v_R)$ 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. 2(c)). Furthermore, let $Z$ be the length of $C \setminus \{e, e', e_L, e_R\}$. Now consider the potential edges $(u, v_R)$, $(v, v_L)$, $(u', u_R)$, and $(v', u_L)$. 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 $(u, v_R)$ be that edge (the construction is fully symmetric). We can now rotate $e$ to $(u, v_R)$, then to $(u', v_R)$, and finally to $e'$. As each part of $C$ has length at most $2\,\mathrm{OPT}/3$, we get that $|(u', v_R)| \leq \mathrm{OPT}/3 + |e|$ by construction. Furthermore we have that $\mathrm{OPT} = |e| + |e_L| + |e_R| + Z$. Thus, $|(u, v_R)| \leq |e| + |e_L|/2 + |e_R|/2 + Z/4 = \mathrm{OPT}/3 + 2|e|/3 + |e_L|/6 + |e_R|/6 - Z/12$. Since $e$ needs to be removed to update the EMST, it must be the longest edge in $C$. Therefore $|(u, v_R)| \leq \mathrm{OPT}/3 + |e|$, 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 7.** *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 $|e| = |e'| = 1$. As a result, the distance between an endpoint of $e$ and the left or right corner of the diamond is $2 - \frac{1}{2}\sqrt{2}$. 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 crosses or touches 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 $9 - 2\sqrt{2}$.




To force the update from $e$ to $e'$ in this configuration, we can use the following motion. The points start at the endpoints of $e$ and move with constant speeds to a position where the points are evenly spread around the left and right corner of the diamond. Then the points move with constant speeds to the endpoints

of $e'$. The argument above still implies that we need edges of total length at least 2 intersecting the vertical diagonal of the diamond at some point during the motion. On the other hand, $OPT \leq 9 - 2\sqrt{2}$ throughout the motion. Thus $\rho_{\mathrm{TS}}(\mathrm{EMST}, \mathcal{T}_{\mathcal{I}}, \mathcal{T}_{\mathcal{S}}) \geq \frac{10-2\sqrt{2}}{9-2\sqrt{2}} \approx 1.162$. □

## 5 Lipschitz stability

The major drawback of topological stability analysis is that it still does not fully capture stable behavior; the algorithm must be continuous, but we can still make many changes to the solution in an arbitrarily small time step. In Lipschitz stability analysis we additionally limit how fast the solution can change.

### 5.1 Lipschitz stability analysis

Let $\Pi$ be an optimization problem with input instances $\mathcal{I}$, solutions $\mathcal{S}$, and optimization function $f$. To quantify how fast a solution changes as the input changes, we need to specify metrics $d_{\mathcal{I}}$ and $d_{\mathcal{S}}$ on $\mathcal{I}$ and $\mathcal{S}$, respectively. An algorithm $\mathcal{A}\colon \mathcal{I} \to \mathcal{S}$ is *$K$-Lipschitz stable* if for any $I, I' \in \mathcal{I}$ we have that $d_{\mathcal{S}}(\mathcal{A}(I), \mathcal{A}(I')) \leq K d_{\mathcal{I}}(I, I')$. We are then again interested in the approximation ratio of any $K$-Lipschitz stable algorithm with respect to OPT. That is, we are interested in the ratio

$$\rho_{\mathrm{LS}}(\Pi, K, d_{\mathcal{I}}, d_{\mathcal{S}}) = \inf_{\mathcal{A}} \sup_{I \in \mathcal{I}} \frac{f(I, \mathcal{A}(I))}{f(I, \mathrm{OPT}(I))} \tag{4}$$

where the infimum is taken over all $K$-Lipschitz stable algorithms. It is easy to see that $\rho_{\mathrm{LS}}(\Pi, K, d_{\mathcal{I}}, d_{\mathcal{S}})$ is lower bounded by $\rho_{\mathrm{TS}}(\Pi, \mathcal{T}_{\mathcal{I}}, \mathcal{T}_{\mathcal{S}})$ for the corresponding topologies $\mathcal{T}_{\mathcal{I}}$ and $\mathcal{T}_{\mathcal{S}}$ of $d_{\mathcal{I}}$ and $d_{\mathcal{S}}$, respectively. As already mentioned in Section 2, analyses of this type are often quite hard. First, we often need to be very careful when choosing the metrics $d_{\mathcal{I}}$ and $d_{\mathcal{S}}$, as they should behave similarly with respect to scale. For example, let the input consist of a set of points in the plane and let $cI$ for $I \in \mathcal{I}$ be the instance obtained by scaling all coordinates of the points in $I$ by the factor $c$. Now assume that $d_{\mathcal{I}}$ depends linearly on scale, that is $d_{\mathcal{I}}(cI, cI') \sim c d_{\mathcal{I}}(I, I')$, and that $d_{\mathcal{S}}$ is independent of scale. Then, for some fixed $K$, we can reduce the effective speed of any $K$-Lipschitz stable algorithm arbitrarily by scaling down the instances sufficiently, rendering the analysis meaningless. We further need to be careful with discrete solution spaces. However, using the flip graphs as mentioned in Section 4 we can extend a discrete solution space to a continuous space by including the edges.

Typically it will be hard to fully describe $\rho_{\mathrm{LS}}(\Pi, K, d_{\mathcal{I}}, d_{\mathcal{S}})$ as a function of $K$. However, it may be possible to obtain interesting results for certain values of $K$. One value of interest is the value of $K$ for which the approximation ratio equals or approaches the approximation ratio of the corresponding topological stability analysis. Another potential value of interest is the value of $K$ below which any $K$-Lipschitz stable algorithm performs asymptotically as bad as a constant algorithm always computing the same solution regardless of instance.

### 5.2 Lipschitz stability of EMSTs

We use the same setting of the kinetic EMST problem as in Section 4.2, except that, instead of topologies, we specify metrics for $\mathcal{I}$ and $\mathcal{S}$. For $d_{\mathcal{I}}$ we can simply use the metric in Equation 2, which implies that points move with a bounded speed. For $d_{\mathcal{S}}$ we use a metric inspired by the edge slides of Section 4.2. To that end, we need to define how long a particular edge slide takes, or equivalently, how "far" an edge slide is. To make sure that $d_{\mathcal{I}}$ and $d_{\mathcal{S}}$ behave similarly with respect to scale, we let $d_{\mathcal{S}}$ measure the distance the sliding endpoint has traveled during an edge slide. However, this creates an interesting problem: the edge on which the endpoint is sliding may be moving and stretching/shrinking during the operation. This influences how long it takes to perform the edge slide. We need to be more specific: (1) as the points are moving, the relative position (between 0 and 1 from starting endpoint to finishing endpoint) of a sliding endpoint is maintained without cost in $d_{\mathcal{S}}$, and (2) $d_{\mathcal{S}}$ measures the difference in relative position multiplied by the length $L(t)$ of the edge on which the endpoint is sliding. More tangibly, an edge slide performed by a $K$-Lipschitz stable algorithm can be performed in $t^*$ time such that $\int_0^{t^*} \frac{K}{L(t)} \mathrm{d}t = 1$, where $L(t)$ describes the length of the edge on which the endpoint slides as a function of time. Finally, the optimization function $f$ simply computes a linear interpolation of the cost on the edges of the flip graph defined by edge slides.

We now give an upper bound on $K$ below which any $K$-Lipschitz stable algorithm for kinetic EMST performs asymptotically as bad as any fixed tree. Given the complexity of the problem, our bound is fairly crude. We state it anyway to demonstrate the use of our framework, but we believe that a stronger bound exists. Note that any spanning tree is an $O(n)$-approximation of the EMST.

**Lemma 8.** *Let $d_{\mathcal{S}}$ be the metric for edge slides, then $\rho_{\mathrm{LS}}(EMST, \frac{c}{\log n}, d_{\mathcal{I}}, d_{\mathcal{S}}) = \Omega(n)$ for a small enough constant $c > 0$, where $n$ is the number of points.*

## 6 Conclusion

We presented a framework for algorithm stability, which includes three types of stability analysis, namely event stability, topological stability, and Lipschitz stability. We also demonstrated the use of this framework by applying the different types of analysis to the kinetic EMST problem, deriving new interesting results. We believe that, by providing different types of stability analysis with increasing degrees of complexity, we make stability analysis for algorithms more accessible, and make it easier to formulate interesting open problems on algorithm stability.

However, the framework that we presented does not (yet) give a complete picture: we do not consider the algorithmic aspect of stability. For example, if we already know how the input will change over time, can we efficiently compute a stable function of solutions over time that is optimal with regard to solution quality? Or, in a more restricted sense, can we efficiently compute the one solution that is best for all inputs over time? Even in the black-box model we can consider

designing efficient algorithms that are $K$-Lipschitz stable and perform well with regard to solution quality. We leave such problems for future work.

## References


1. O. Aichholzer, F. Aurenhammer, and F. Hurtado. Sequences of spanning trees and a fixed tree theorem. *Comput. Geom. Theory Appl.*, 21(1-2):3–20, 2002.
2. A. Bacciotti and L. Rosier. *Liapunov Functions and Stability in Control Theory.* Springer, 2nd edition, 2006.
3. J. Basch, L. J. Guibas, and J. Hershberger. Data structures for mobile data. *Journal of Algorithms*, 31(1):1–28, 1999.
4. K. Been, M. Nöllenburg, S.-H. Poon, and A. Wolff. Optimizing active ranges for consistent dynamic map labeling. *Comput. Geom. Theory Appl.*, 43(3):312–328, 2010.
5. O. Bousquet and A. Elisseeff. Stability and generalization. *Journal of Machine Learning Research*, 2:499–526, 2002.
6. 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. *Comput. Geom. Theory Appl.*, 36(2):117–130, 2007.
7. D. Cohen-Steiner, H. Edelsbrunner, and J. Harer. Stability of persistence diagrams. In *Proc. 21st Symp. Comput. Geom.*, pages 263–271, 2005.
8. M. de Berg, M. Roeloffzen, and B. Speckmann. Kinetic 2-centers in the black-box model. In *Proc. 29th Symp. Comput. Geom.*, pages 145–154, 2013.
9. S. Durocher and D. Kirkpatrick. The steiner centre of a set of points: Stability, eccentricity, and applications to mobile facility location. *Int. J. Comput. Geom. Appl.*, 16(04):345–371, 2006.
10. S. Durocher and D. Kirkpatrick. Bounded-velocity approximation of mobile Euclidean 2-centres. *Int. J. Comput. Geom. Appl.*, 18(03):161–183, 2008.
11. J. Erickson. Dense point sets have sparse Delaunay triangulations or ... but not too nasty. *Discrete Comput. Geom.*, 33(1):83–115, 2005.
12. W. Goddard and H. C. Swart. Distances between graphs under edge operations. *Discrete Mathematics*, 161(1-3):121–132, 1996.
13. 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.
14. N. J. Higham. *Accuracy and Stability of Numerical Algorithms.* SIAM, 2002.
15. N. Katoh, T. Tokuyama, and K. Iwano. On minimum and maximum spanning trees of linearly moving points. In *Proc. 33rd Symp. Found. Comp. Sci.*, pages 396–405, 1992.
16. R. M. Kitchin. Cognitive maps: What are they and why study them? *Journal of Environmental Psychology*, 14(1):1–19, 1994.
17. W. Markoff. Über Polynome, die in einem gegebenen Intervalle möglichst wenig von Null abweichen. *Mathematische Annalen*, 77(2):213–258, 1916.
18. C. Monma and S. Suri. Transitions in geometric minimum spanning trees. *Discrete Comput. Geom.*, 8(3):265–293, 1992.
19. D. A. Spielman and S.-H. Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *Journal of the ACM*, 51(3):385–463, 2004.