A framework for algorithm stability

A Framework for Algorithm Stability

Wouter Meulemans[1], Bettina Speckmann[1], Kevin Verbeek[1], and Jules Wulms[1]

1 **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

Abstract

We say that an algorithm is *stable* if small changes in the input result in small changes in the output. Algorithm stability plays an important role when analyzing and visualizing time-varying data. However, so far, there are only few theoretical results on the stability of algorithms, possibly due to a lack of theoretical analysis tools. 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.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems: Geometrical problems and computations

Keywords and phrases Stability framework, stability analysis, time-varying data, minimum spanning tree

1 Introduction

With recent advances in technology, vast amounts of *time-varying data* are being processed and analyzed on a daily basis. Time-varying data play an important role in a many different application areas, such as computer graphics, robotics, simulation, and visualization. There is a great need for algorithms that can operate efficiently on time-varying data and that can offer guarantees on the quality of the results. A specific relevant subset of time-varying data are *motion data*: geometric time-varying data. To deal with the challenges of motion data, Basch *et al.* [4] introduced the *kinetic data structures* (KDS) framework in 1999. Kinetic data structures are data structures that efficiently maintain a structure on a set of moving objects. The framework has sparked a large amount of research and has resulted in many efficient algorithms for motion data. However, one aspect of algorithms for time-varying data has received little attention in the theoretical computer science community so far: *stability*. Whenever analysis results on time-varying data need to be communicated to humans, for example via visual representations, it is important that these results are stable: small changes in the data result in small changes in the output. Sudden changes in the visual representation of data disrupt the *mental map* [19] of the user and prevent the recognition of temporal patterns. Stability also plays a role if changing the result is costly in practice, for example in physical network design, where frequent total overhauls of the network are simply infeasible.

We would hence like to argue that, next to running time and solution quality, stability could and should be a third important criterion when analyzing and comparing algorithms. Furthermore, it is particularly interesting to study the tradeoffs that can be made among these three criteria. However, currently there are no good tools to formally analyze or measure the stability of an algorithm, and as a result, the tradeoff between running time, solution quality, and stability is poorly understood for most problems.

Results and organization. We present a framework to measure and analyze the stability of 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 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 Appendix A.

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 [5, 14, 13, 22], 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, 10] 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 [8] consider the tradeoff between the solution quality of Euclidean 2-centers and a bound on the velocity with which they can move. Furthermore, De Berg *et al.* [7] show similar results in the black-box KDS model. One can argue that the KDS framework [16] 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 quality of the results to reduce the number of external events.

Kinetic Euclidean minimum spanning trees and related structures have been studied extensively. Katoh *et al.* [18] 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. Rahmati *et al.* [23] present a kinetic data structure for EMSTs in the plane that processes $O(n^3 \beta^2_{2s+2}(n) \log n)$ events in the worst case, where s is a measure for the complexity of the point trajectories and $\beta_s(n)$ is an extremely slow-growing function. The best known lower bound for external events of EMSTs in d dimensions is $\Omega(n^d)$ [21]. Since the EMST is a subset of the Delaunay triangulation, we can also consider to kinetically maintain the Delaunay triangulation instead. Fu and Lee [11], and Guibas *et al.* [17] show that the Delaunay triangulation undergoes $O(n^2 \lambda_{s+2}(n))$ external events (near-cubic), where $\lambda_s(n)$ is the maximum length of an (n, s)-Davenport-Schinzel sequence [28]. On the other hand, the best lower bound for external events of the Delaunay triangulation is only $\Omega(n^2)$ [28]. Rubin improves the upper bound to $O(n^{2+\varepsilon})$, for any $\varepsilon > 0$, if the number of degenerate events is limited [26], or if the points move along a straight line with unit speed [27]). Agarwal *et al.* [1] also consider a more stable version of the Delaunay triangulation, which undergoes at most a nearly quadratic number of external events. However, external events for EMSTs do not necessarily coincide with external events of the Delaunay triangulation [24]. To further reduce the number of external events, we can consider approximations of the EMST, for example via spanners or well-separated pair decompositions [3]. However, kinetic t-spanners

already undergo $\Omega(\frac{n^2}{t^2})$ external events [12]. 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, we can formulate this concept generically as follows. 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[1] 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. As with the distortion of metric embeddings, there are many other ways to define the stability of an algorithm given the metrics, but the above definition is sufficient 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

[1] A metric would typically be the most suitable solution, but any dissimilarity function is sufficient.

space $\mathcal{S}$ that captures stable behavior. Surprisingly, even though we completely 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 some results may be obtained for values that are sufficiently small or large.

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.

3 Event stability

The simplest and most intuitive form of stability is event stability. Similarly to the number of external events in KDSs, event stability captures only how often the solution changes.

3.1 Event stability analysis

Let Π 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. 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.

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 generally assume that the functions x_i are polynomials with bounded degree s.

To properly 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.

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

This metric simply measures the distance traveled of the point that traveled the farthest. 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. 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 how often we need to recompute the EMST, and how well a k-optimal solution approximates the EMST.

Number of events. To derive an upper bound on 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** ([20]). *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.** *Let P be a kinetic point set with polynomial trajectories $x_i(t) \in [0, 1]^d$ ($t \in [0, T]$) of degree at most s, then we need only $O(\frac{s^2}{k})$ changes to maintain a k-optimal solution.*

Proof. By Lemma 1 the velocity of any point is at most s^2/T. Now assume that we have computed an optimal solution S for some time t. The solution S remains k-optimal until

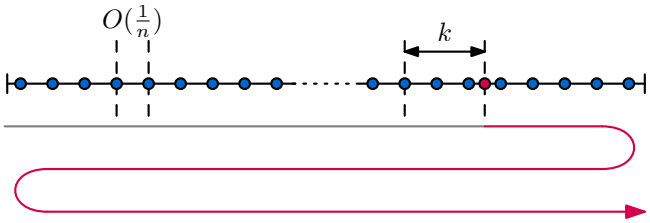

Figure 1 An instance where $\Omega(\frac{s}{k})$ events are triggered. The blue points are stationary, while the red point moves along a trajectory of degree 3. The trajectory is shown as an arrow along the 1D space, where the gray part has already been traversed.

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. ◀

Next we show that this upper bound is tight up to a factor of s, using *Chebyshev polynomials* of the first kind [25]. A Chebyshev polynomial of degree s with range $[0, 1]$ and domain $[0, T]$ will pass through the entire range exactly s times.

▶ **Lemma 3.** *Let P be a kinetic point set with n points with polynomial trajectories $x_i(t) \in [0, 1]^d$ ($t \in [0, T]$) of degree at most s, then we need $\Omega(\min(\frac{s}{k}, sn))$ changes in the worst case to maintain a k-optimal solution.*

Proof. We can restrict ourselves to $d = 1$. Let p_1 move along a Chebyshev polynomial of degree s, and let the remaining points be stationary and placed equidistantly along the interval $[0, 1]$. As soon as p_1 meets one of the other points, then p_1 can travel at most k units before the solution is no longer k-optimal (see Fig.1). Therefore, p_1 moving through the entire interval requires $\Omega(\min(1/k, n))$ changes to the solution. Doing so s times gives the desired bound. ◀

It is important to notice that this behavior is fairly special for polynomial trajectories. If we allow more general trajectories, then this bound on the number of changes breaks down.

▶ **Lemma 4.** *Let P be a kinetic point set with n points with pseudo-algebraic trajectories $x_i(t) \in [0, 1]^d$ ($t \in [0, T]$) of degree at most s, then we need $\Omega(\min(\frac{sn}{k}, sn^2))$ changes in the worst case to maintain a k-optimal solution.*

Proof. We can restrict ourselves to $d = 1$. Any two pseudo-algebraic trajectories of degree at most s can cross each other at most s times. We make $n/2$ points stationary and place them equidistantly along the interval $[0, 1]$. The other $n/2$ points follow trajectories that take them through the entire interval s times, in such a way that every point moves through the entire interval completely before another point does so. The resulting trajectories are clearly pseudo-algebraic, and each time a point moves through the entire interval it requires $\Omega(\min(1/k, n))$ changes to the solution. As a result, the total number of changes is $\Omega(\min(\frac{sn}{k}, sn^2))$. ◀

We can show the same lower bound for algebraic trajectories of degree at most s, but this is slightly more involved. The result can be found in Appendix A.

Approximation factor. We cannot expect k-optimal solutions to be a good approximation of optimal EMSTs 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 [9]. 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 Δ_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 5.** *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 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

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

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 so-called flip graphs, for example with edge flips for triangulations or rotations in rooted binary trees. The graph topology of the flip graph can then be used in the above definition. To make this proper, we also need to define f on the edges of this graph, but a suitable choice can easily be made based on the problem itself; often a linear interpolation on the edges is sufficient.

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 [2, 15]. For edge slides we give tight bound on $\mathrm{TS}(\mathrm{EMST}, \mathcal{T}_\mathcal{I}, \mathcal{T}_\mathcal{S})$, and for edge rotations we provide upper and lower bounds.

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. Since this operation is very local, we consider it to be stable. Note that after every edge slide the tree must still be connected. Using Lemmata 6 and 7 we show that $\mathrm{TS}(\mathrm{EMST}, \mathcal{T}_\mathcal{I}, \mathcal{T}_\mathcal{S}) = \frac{3}{2}$ for edge slides.

▶ **Lemma 6.** *If $\mathcal{T}_\mathcal{S}$ is the topology on $\mathcal{S}$ defined by edge slides, then* $\mathrm{TS}(\mathit{EMST}, \mathcal{T}_\mathcal{I}, \mathcal{T}_\mathcal{S}) \leq \frac{3}{2}$.

Proof. Consider a point in time where the EMST has to be updated by removing an edge e and inserting an edge e', where $|e| = |e'|$. 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). 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 $|C| - 2|e| + |x| = \mathrm{OPT} - |e| + |x| \leq \frac{3}{2}\,\mathrm{OPT}$. ◀

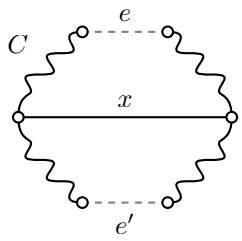

Figure 2 x is the longest edge during an edge slide from e to e'.

▶ **Lemma 7.** *If $\mathcal{T}_\mathcal{S}$ is the topology on $\mathcal{S}$ defined by edge slides, then, for any $\varepsilon > 0$,* $\mathrm{TS}(\mathit{EMST}, \mathcal{T}_\mathcal{I}, \mathcal{T}_\mathcal{S}) \geq \frac{3}{2} - \varepsilon$.

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 and e' share an endpoint v. Let the remainder of the EMST be very thin and elongated, as shown in Figure 3, such that the distance between v

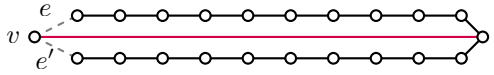

Figure 3 An instance that shows how an edge slide from edge e to edge e' will at some point be in the red configuration. This configuration is a $(\frac{3}{2} - \varepsilon)$-approximation of the EMST.

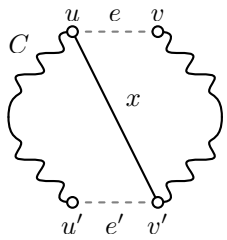

Figure 4 If one part of C is small enough, then we can rotate one endpoint of e directly to one endpoint of e'.

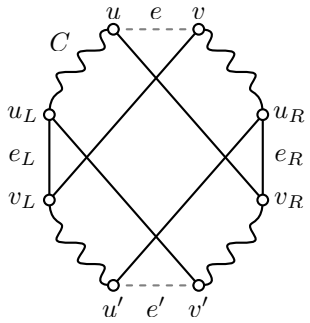

Figure 5 The potential intermediate edges when rotating edge e to e'.

and its farthest point is $\mathrm{OPT}/2 - \varepsilon$, 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. Since any edge slide from e to e' includes all edges between v and all other points, we get that, for any $\varepsilon > 0$, $\mathrm{TS}(\mathrm{EMST}, \mathcal{T}_\mathcal{I}, \mathcal{T}_\mathcal{S}) \geq \frac{3}{2} - \varepsilon$. If the algorithm does not decide to slide edge e to e', then we can gradually move the top points towards the rightmost point. The argument now remains the same as the EMST is shrinking. ◀

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. Using Lemmata 8 and 9 we show that $\frac{5}{4} \leq \mathrm{TS}(\mathrm{EMST}, \mathcal{T}_\mathcal{I}, \mathcal{T}_\mathcal{S}) \leq \frac{4}{3}$ for edge rotations.

▶ **Lemma 8.** *If $\mathcal{T}_\mathcal{S}$ is the topology on $\mathcal{S}$ defined by edge rotations, then* $\mathrm{TS}(\mathit{EMST}, \mathcal{T}_\mathcal{I}, \mathcal{T}_\mathcal{S}) \leq \frac{4}{3}$.

Proof. Consider a point in 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 will split 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. 4). 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. 5). 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 $\mathrm{TS}(\mathrm{EMST}, \mathcal{T}_\mathcal{I}, \mathcal{T}_\mathcal{S}) \leq \frac{4}{3}$. ◀

▶ **Lemma 9.** *If $\mathcal{T}_\mathcal{S}$ is the topology on $\mathcal{S}$ defined by edge rotations, then, for any $\varepsilon > 0$, $\mathrm{TS}(\mathit{EMST}, \mathcal{T}_\mathcal{I}, \mathcal{T}_\mathcal{S}) \geq \frac{5}{4} - \varepsilon$.*

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 diamond shape as shown in Figure 6. When rotating from e to e', we must in some step have an edge that connects the left top of the diamond with the right bottom (or the right top with the left bottom). The length of this edge x must be at least the length of a side of the diamond. By using enough points and making e and e' short enough, we can ensure that, for any $\varepsilon > 0$, $|x| \geq \mathrm{OPT}/4 - \varepsilon$, where OPT is the length of the EMST. If e is short enough, this implies that, for any $\varepsilon > 0$, $\mathrm{TS}(\mathrm{EMST}, \mathcal{T}_\mathcal{I}, \mathcal{T}_\mathcal{S}) \geq \frac{5}{4} - \varepsilon$. If the algorithm does not decide to rotate edge e to e', then we can gradually move the right top points towards the rightmost point, and the left top points towards the leftmost point. The argument now remains the same as the EMST is shrinking. ◀

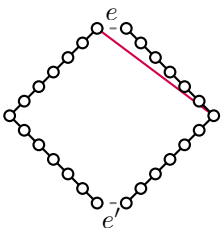

Figure 6 Lower bound construction for edge rotations.

Unfortunately, we do not obtain tight bounds for the edge rotations. This is caused by the fact that our upper bound only uses the triangle inequality, and not any properties specific to Euclidean distance. Indeed, if we would allow general metrics, then the upper bound is tight. The lower bound is realized by the graph metric on a 4-cycle, when the middle edge of an EMST is replaced by the opposite edge.

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 Π 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

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

where the infimum is taken over all K-Lipschitz stable algorithms. It is easy to see that $\mathrm{LS}(\Pi, K, d_\mathcal{I}, d_\mathcal{S})$ is lower bounded by $\mathrm{TS}(\Pi, \mathcal{T}_\mathcal{I}, \mathcal{T}_\mathcal{S})$ 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 $d_{\mathcal{I}}(cI, cI') \sim cd_{\mathcal{I}}(I, I')$, 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 $\mathrm{LS}(\Pi, K, d_{\mathcal{I}}, d_{\mathcal{S}})$ 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 from 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 (see Fig. 7). 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 provide it anyway to demonstrate the use of our framework, but we believe that a stronger bound exists. First we show the asymptotic approximation ratio of any spanning tree.

▶ **Lemma 10.** *Any spanning tree on a set of n points P is an $O(n)$-approximation of the EMST.*

▶ **Lemma 11.** *Let $d_{\mathcal{S}}$ be the metric for edge slides, then $\mathrm{LS}(EMST, \frac{c}{\log n}, d_{\mathcal{I}}, d_{\mathcal{S}}) = \Omega(n)$ for a small enough constant $c > 0$, where n is the number of points.*

Proof. Consider the instance where n points are placed equidistantly vertically above each other with distance $1/n$ between two consecutive points. Now let $\mathcal{A}$ be any $(c/\log n)$-Lipschitz stable algorithm for the kinetic EMST problem and let T be the tree computed by $\mathcal{A}$ on this point set. We now color the points red and blue based on a 2-coloring of T. We then

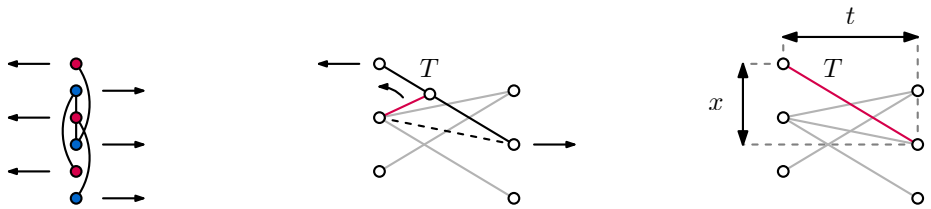

Figure 7 An instance where any K-Lipschitz stable algorithm performs badly: Starting from a solution on equidistantly placed points, the red and blue points move in opposite directions (left). The endpoint of the red edge slides over stretching edge T (middle). The length of edge T at time t is $\sqrt{x^2+t^2}$ (right).

move the red points to the left by $\frac{1}{2}$ and the blue points to the right by $\frac{1}{2}$ in the time interval $[0,1]$ (see Fig. 7 left). This way every edge of T will be stretched to a length of $\Omega(1)$ and thus the length of T will be $\Omega(n)$. On the other hand, the length of the EMST in the final configuration is $\mathrm{OPT} = O(1)$. Therefore, we must perform an edge slide (see Fig. 7 middle). However, we show that $\mathcal{A}$ cannot complete any edge slide. Consider any edge of T and let x be the initial (vertical) distance between its endpoints. Then the length of this edge can be described as $L(t) = \sqrt{x^2+t^2}$ (see Fig. 7 right). Now assume that we want to slide an endpoint over this edge. To finish this edge slide before $t = 1$, we require that $\int_0^1 \frac{c}{\log n\sqrt{x^2+t^2}}\mathrm{d}t \geq 1$. This solves to $c\log(1/x+\sqrt{1+1/x^2}) \geq \log n$. However, since $x \geq 1/n$, we get that $c\log(1/x+\sqrt{1+1/x^2}) \leq c\log(n+\sqrt{1+n^2}) < \log n$ for c small enough. Finally, since one edge slide can reduce the length of only one edge to $o(1)$, the cost of the solution at $t = 1$ computed by $\mathcal{A}$ is $\Omega(n)$. Thus, $\mathrm{LS}(\mathrm{EMST}, \frac{c}{\log n}, d_{\mathcal{I}}, d_{\mathcal{S}}) = \Omega(n)$ for a small enough constant $c > 0$. ◀

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 with regard to algorithm stability.

However, the framework that we have presented is not designed to offer a complete picture on algorithm stability. In particular, 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.

Acknowledgements. 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.

References

1 Pankaj K. Agarwal, Jie Gao, Leonidas J. Guibas, Haim Kaplan, Natan Rubin, and Micha Sharir. Stable Delaunay graphs. *Discrete & Computational Geometry*, 54(4):905–929, 2015.

2 Oswin Aichholzer, Franz Aurenhammer, and Ferran Hurtado. Sequences of spanning trees and a fixed tree theorem. *Computational Geometry: Theory and Applications*, 21(1-2):3–20, 2002.

3 Sunil Arya and David M. Mount. A fast and simple algorithm for computing approximate Euclidean minimum spanning trees. In *Proc. 27th Symposium on Discrete Algorithms*, pages 1220–1233, 2016.

4 Julien Basch, Leonidas J. Guibas, and John Hershberger. Data structures for mobile data. *Journal of Algorithms*, 31(1):1–28, 1999.

5 Ken Been, Martin Nöllenburg, Sheung-Hung Poon, and Alexander Wolff. Optimizing active ranges for consistent dynamic map labeling. *Computational Geometry: Theory and Applications*, 43(3):312–328, 2010.

6 Peter Brass, Eowyn Cenek, Cristian A. Duncan, Alon Efrat, Cesim Erten, Dan P. Ismailescu, Stephen G. Kobourov, Anna Lubiw, and Joseph S.B. Mitchell. On simultaneous planar graph embeddings. *Computational Geometry: Theory and Applications*, 36(2):117–130, 2007.

7 Mark de Berg, Marcel Roeloffzen, and Bettina Speckmann. Kinetic 2-centers in the black-box model. In *Proc. 29th Symposium on Computational Geometry*, pages 145–154, 2013.

8 Stephane Durocher and David Kirkpatrick. Bounded-velocity approximation of mobile Euclidean 2-centres. *International Journal of Computational Geometry & Applications*, 18(03):161–183, 2008.

9 Jeff Erickson. Dense point sets have sparse Delaunay triangulations or "... but not too nasty". *Discrete & Computational Geometry*, 33(1):83–115, 2005.

10 Fabrizio Frati, Michael Kaufmann, and Stephen G. Kobourov. Constrained simultaneous and near-simultaneous embeddings. *Journal of Graph Algorithms and Applications*, 13(3):447–465, 2009.

11 Jyh-Jong Fu and Richard C.T. Lee. Voronoi diagrams of moving points in the plane. *International Journal of Computational Geometry & Applications*, 1(01):23–32, 1991.

12 Jie Gao, Leonidas J. Guibas, and An Nguyen. Deformable spanners and applications. *Computational Geometry: Theory and Applications*, 35(1-2):2–19, 2006.

13 Andreas Gemsa, Martin Nöllenburg, and Ignaz Rutter. Sliding labels for dynamic point labeling. In *Proc. 23rd Canadian Conference on Computational Geometry*, pages 205–210, 2011.

14 Andreas Gemsa, Martin Nöllenburg, and Ignaz Rutter. Consistent labeling of rotating maps. *Journal of Computational Geometry*, 7(1):308–331, 2016.

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

16 Leonidas J. Guibas. Kinetic data structures. In Dinesh P. Mehta and Sartaj Sahni, editors, *Handbook of Data Structures and Applications*, pages 23–1–23–18. Chapman and Hall/CRC, 2004.

17 Leonidas J. Guibas, Joseph S.B. Mitchell, and Thomas Roos. Voronoi diagrams of moving points in the plane. In *Proc. 17th International Workshop on Graph-Theoretic Concepts in Computer Science*, LNCS 570, pages 113–125, 1992.

18 Naoki Katoh, Takeshi Tokuyama, and Kazuo Iwano. On minimum and maximum spanning trees of linearly moving points. In *Proc. 33rd Symposium on Foundations of Computer Science*, pages 396–405, 1992.

19 Robert M. Kitchin. Cognitive maps: What are they and why study them? *Journal of Environmental Psychology*, 14(1):1–19, 1994.

20 Wladimir Markoff. Über Polynome, die in einem gegebenen Intervalle möglichst wenig von Null abweichen. *Mathematische Annalen*, 77(2):213–258, 1916.

21 Clyde Monma and Subhash Suri. Transitions in geometric minimum spanning trees. *Discrete & Computational Geometry*, 8(3):265–293, 1992.

22 Martin Nöllenburg, Valentin Polishchuk, and Mikko Sysikaski. Dynamic one-sided boundary labeling. In *Proc. 18th Conference on Advances in Geographic Information Systems*, pages 310–319, 2010.

23 Zahed Rahmati, Mohammad Ali Abam, Valerie King, Sue Whitesides, and Alireza Zarei. A simple, faster method for kinetic proximity problems. *Computational Geometry: Theory and Applications*, 48(4):342–359, 2015.

24 Zahed Rahmati and Alireza Zarei. Kinetic Euclidean minimum spanning tree in the plane. *Journal of Discrete Algorithms*, 16:2–11, 2012.

25 Theodore J. Rivlin. *The Chebyshev Polynomials*, volume 40 of *Pure and Applied Mathematics*. John Wiley & Sons, 1974.

26 Natan Rubin. On topological changes in the Delaunay triangulation of moving points. *Discrete & Computational Geometry*, 49(4):710–746, 2013.

27 Natan Rubin. On kinetic Delaunay triangulations: A near-quadratic bound for unit speed motions. *Journal of the ACM*, 62(3):25, 2015.

28 Micha Sharir and Pankaj K. Agarwal. *Davenport-Schinzel sequences and their geometric applications*. Cambridge University Press, 1995.

A Omitted proofs

▶ **Lemma 12.** *Let P be a kinetic point set with n points with algebraic trajectories $x_i(t) \in [0,1]^d$ ($t \in [0,T]$) of degree at most s, then we need $\Omega(\min(\frac{sn}{k}, sn^2))$ changes in the worst case to maintain a k-optimal solution.*

Proof. We can restrict ourselves to $d = 1$. We make $n/2$ points stationary and place them equidistantly along the interval $[0,1]$. The other $n/2$ points follow trajectories that take them through the entire interval $s/4$ times, in such a way that every point moves through the entire interval completely before another point does so. The trajectory of a non-stationary point p_i is $x_i(t) = \sum_{j=0}^{s/4} \frac{1}{(t-10\cdot j-10\cdot i\cdot s/4)^4+1}$. The trajectory consists of $s/4$ moves through the stationary points, one such move every 10 time units (see Fig. 8). The i-th point will be finished $10*s/4$ time units after it starts its first move through the stationary points, while the $(i+1)$-st point starts 10 units after the i-th point finishes. The resulting trajectories are clearly algebraic, and each time a point moves through the entire interval it requires $\Omega(\min(1/k, n))$ changes to the solution. As a result, the total number of changes is $\Omega(\min(\frac{sn}{k}, sn^2))$. ◀

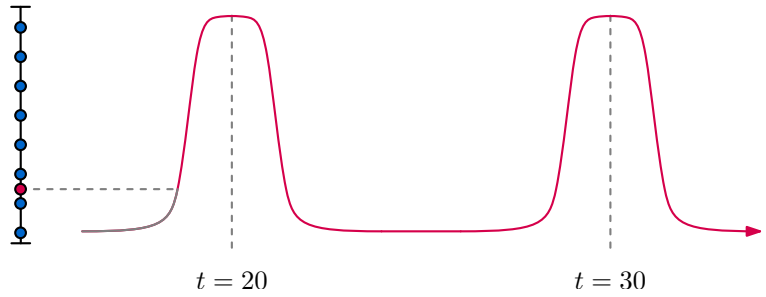

Figure 8 An instance where the red point moves along the trajectory $x_1(t) = \sum_{j=0}^{2} \frac{1}{(t-10\cdot j-20)^4+1}$. The blue points are stationary, while the red point moves along a trajectory of degree 8. The red point is the second point to start moving through the blue points (at $t = 20$). The trajectory is shown as an arrow along the 1D space, where the gray part has already been traversed.

▶ **Lemma 10.** *Any spanning tree on a set of n points P is an $O(n)$-approximation of the EMST.*

Proof. Let T be an EMST on point set P with total edge length OPT. Additionally let $u, v \in P$, and observe that the path along T from u to v is at least the Euclidean distance between u and v, $d(u,v) \leq \text{PATH}_T(u,v)$. Furthermore, any path along an EMST is at most as long as the total length of an EMST, $\text{PATH}_T(u,v) \leq \text{OPT}$. If we now take an arbitrary spanning tree T' on the same point set P, then we know that each edge (u',v') in this spanning tree has at most length $d(u',v') \leq \text{PATH}_T(u',v') \leq \text{OPT}$. Since T' has $n-1$ edges, its total length is $O(n) \cdot \text{OPT}$. ◀