International Journal of Computational Geometry & Applications

DYNAMIC POINT LABELING IS STRONGLY PSPACE-COMPLETE

KEVIN BUCHIN, DIRK H.P. GERRITS
Department of Mathematics and Computer Science
Technische Universiteit Eindhoven
Den Dolech 2, 5600 MB Eindhoven, The Netherlands
k.a.buchin@tue.nl, dirk@dirkgerrits.com

Received (received date)
Revised (revised date)
Communicated by (Name)

An important but strongly NP-hard problem in automated cartography is how to best place textual labels for point features on a static map. We examine the complexity of various generalizations of this problem for dynamic and/or interactive maps. Specifically, we show that it is strongly PSPACE-complete to decide whether there is a smooth dynamic labeling (function from time to static labelings) when the points move, when points are added and removed, or when the user pans, rotates, and/or zooms their view of the points. In doing so we develop a framework from which a wide variety of labeling hardness results can be obtained, including (next to the PSPACE-hardness results) both known and new results on the NP-hardness of static labeling.

1. Introduction

Map labeling involves associating textual *labels* with certain *features* on a map such as cities (points), roads (polylines), and lakes (polygons). This task takes considerable time to do manually, and for some applications *cannot* be done manually beforehand. In air traffic control, for example, a set of moving points (airplanes) has to be labeled at all times. In interactive maps users may pan, rotate, and/or zoom their view of the map, which may also require relabeling. It is therefore unsurprising that map labeling has attracted considerable algorithmic research (see, for instance, the on-line Map Labeling Bibliography,[1] currently containing 371 references).

Static points. A good labeling for a point set has readable labels and an unambiguous association between the labels and the points. This has been formalized by regarding the labels as axis-aligned rectangles slightly larger than the text they contain, which must be placed without overlap so that each contains the point it labels on its boundary. Not all placements are equally desirable, and as such various *label models* have been proposed which specify allowed positions for the labels (Figure 1). In the *fixed-position models*, every point has a finite number of label

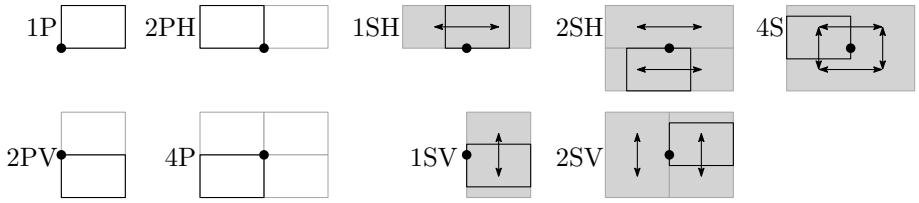

Fig. 1. A label model specifies the allowed positions (shown in gray) for the label of a point. Fixed-position models: the 1-position (1P), 2-position (2PH, 2PV), and 4-position (4P) models. Slider models: the 1-slider (1SH, 1SV), 2-slider (2SH, 2SV), and 4-slider (4S) models.

candidates. In particular, in the 1-, 2-, and 4-position models a subset of 1, 2, or 4 corners of the labels is designated (the same subset for each label). Each label must then have one of its designated corners coincide with the point it labels. The *slider models* generalize this. In the 1-slider models one side of each label is designated, but the label may contain its point anywhere on this side. In the 2-slider models there is a choice between two opposite sides, and in the 4-slider model the label can contain the point anywhere on its boundary.

Ideally, one would label all points with non-intersecting labels, but this is not always possible. Deciding this is a strongly NP-complete problem for the 4-position,[2] 2-slider (Corollary 1 in Section 2.4), and 4-slider[4] models, even if all labels are unit squares. We may deal with this difficulty in several ways. Firstly, we may shrink the labels. The *size-maximization problem* asks to label all points with non-intersecting labels of maximal size. Secondly, we may remove labels. The *(weighted) number-maximization problem* asks to label a maximum-cardinality (maximum-weight) subset of the points with non-intersecting labels of given dimensions. Thirdly, we may allow labels to overlap, but try to keep such occurrences to a minimum. The *free-label-maximization problem* asks for all points to be labeled with labels of given dimensions, maximizing the number of non-intersecting labels. As the decision problem mentioned above is strongly NP-hard for the 4-position, 2-slider, and 4-slider models, these three optimization problems are as well.

Dynamic points. A natural generalization of static point labeling is dynamic point labeling. Here the point set P changes over time, by points being added and removed, and/or by points moving continuously. This can be inherent to the point set (as in air traffic control), or be the result of the user panning, rotating, and zooming (as in interactive maps). Thus, the input is a *dynamic point set* P, which specifies for each point $p \in P$ its arrival and departure times, as well as its trajectory. We seek a *dynamic labeling* $\mathcal{L}$, which for all t assigns a static labeling $\mathcal{L}(t)$ to the static points $P(t)$ present at time t. We would like this assignment to result in continuous label motion, but this is only a reasonable requirement for the 1-slider and 4-slider models. In the fixed-position model each label would have to remain stationary relative to its point, and the 2-slider model would effectively become

a 1-slider model. We want to avoid such a restriction on the range of valid label positions, while still requiring "mostly continuous" label motion. To this end we do allow the label for a point p to "jump" discontinuously from one pposition A to another position B, but only if there is no valid intermediate position C in between A and B in clockwise or counter-clockwise order around p. For the 4-position model we thus allow horizontal and vertical jumps, but no diagonal ones. For a 2-slider model the label must move continuously along the sliders, and may only jump from the endpoint of a slider to the "same" endpoint on the other slider. This ensures that the label motion could have been the result of snap rounding a 4-slider label motion.

For static point labeling, practical heuristic algorithms and theoretical algorithms with guaranteed approximation ratios abound. Dynamic point labeling, however, has seen very few theoretical results. Been et al.[5] studied number-maximization for points under zooming, giving constant-factor approximations for unit-square labels in the 1-position model. Gemsa et al.[6] gave a PTAS for the same problem with rotation instead of zooming. Theoretical treatment of other label models and general point trajectories are sorely missing.

Our results. We believe the relative lack of theoretical results for dynamic point labeling is not due to a lack of attempts. Intuitively, dynamic point labeling should be much harder than its static counterpart. We prove and quantify this intuition. Specifically, we consider the problem of deciding whether there exists a dynamic labeling without intersections for a given dynamic point set. We prove that this is PSPACE-complete for unit-square labels in the 4-position, 2-slider, and 4-slider models, and remains so even if the input is given in unary notation (it is *strongly* PSPACE-complete). This is the case when points are added or removed from the point set, when (some of) the points move, and when the point set is panned, rotated, or zoomed within a finite viewport. Any dynamic generalization of the mentioned static optimization problems is therefore strongly PSPACE-hard in these settings. Additionally, we prove that label-size maximization on dynamic point sets admits no PTAS unless P=PSPACE.

2. Structure of the reduction

To prove PSPACE-hardness of dynamic point labeling, we reduce from *non-deterministic constraint logic (NCL)*,[7] which is an abstract, single-player game. The game board is a *constraint graph*: an undirected graph with non-negative weights on both the vertices and the edges. A *configuration* of the constraint graph specifies an orientation for each of its edges. A configuration is *legal* if and only if each vertex's *inflow* (the summed weight of its incoming edges) is at least its own weight. The outflow of vertices, on the other hand, is not constrained. To make a *move* in this game is to reverse a single edge in a legal configuration such that the resulting configuration is again legal. Hearn and Demaine[7] showed that each of the following

questions is PSPACE-complete.

- *Configuration-to-configuration NCL*: Given two legal configurations $\mathcal{C}_A$ and $\mathcal{C}_B$, is there a sequence of moves transforming $\mathcal{C}_A$ into $\mathcal{C}_B$?
- *Configuration-to-edge NCL*: Given a legal configuration $\mathcal{C}_A$ and an orientation for a single edge e_B, is there a sequence of moves transforming $\mathcal{C}_A$ into a legal configuration $\mathcal{C}_B$ in which e_B has the specified orientation?
- *Edge-to-edge NCL*: Given orientations for edges e_A and e_B, do there exist legal configurations $\mathcal{C}_A$ and $\mathcal{C}_B$, and a sequence of moves transforming the former into the latter, such that e_A has the specified orientation in $\mathcal{C}_A$ and e_B has the specified orientation in $\mathcal{C}_B$?

These decision problems remain PSPACE-complete even for planar, 3-regular constraint graphs consisting only of AND vertices and protected OR vertices. These vertex types are depicted in Figure 2, and are defined as follows. An *AND vertex* has a weight of 2, and its three incident edges have weights 1, 1, 2—see Figure 2(a). To orient the weight-2 edge away from the vertex requires both weight-1 edges to be oriented towards the vertex. An *OR vertex* also has weight 2, as do its three incident edges—see Figure 2(b). Thus at least one edge needs to be oriented towards the vertex at all times. An OR vertex is called *protected* if it has two edges that, because of constraints imposed on them by the rest of the constraint graph, cannot both be directed inward. One way to achieve this is with a triangle of two AND vertices and one OR vertex, as in Figure 2(c). The AND vertices then share a weight-1 edge, meaning at least one of them must have an incoming weight-2 edge to satisfy its inflow constraint. This edge then cannot point into the OR vertex, making it protected. We may actually assume that all OR vertices occur in such triangles, as that is the case in Hearn and Demaine's construction to simulate regular ORs with protected ORs.[7]

As a first step towards proving hardness of dynamic labeling, we will show how to simulate a constraint graph G with a point set P. This will be done in such a way that labelings of P will correspond to configurations of G, and label movements will correspond to changing the orientations of edges in G. In the next section, we will

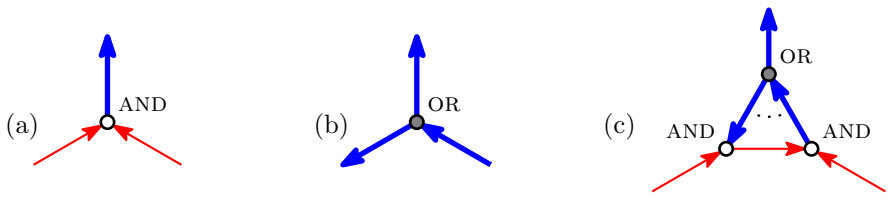

Fig. 2. The vertex types of non-deterministic constraint logic. In this figure, and all that follow, weight-1 edges are drawn as thin, red lines and weight-2 edges are drawn as thick, blue lines. (a) An AND vertex, indicated as a white disk. (b) An OR vertex, indicated as a gray disk. (c) A way to create a protected OR vertex, where the two edges indicated by the dotted arc cannot be directed inward simultaneously.

then show how to use this construction to prove the PSPACE-hardness of various dynamic labeling problems.

2.1. *High-level overview*

At a high level, our construction works as follows. Given a planar constraint graph G with n vertices, we first embed it on a regular grid that has been turned by 45° relative to the coordinate axes. In this embedding G's vertices lie on grid vertices, and its edges form interior-disjoint paths along grid lines, as depicted in the center of Figure 3. From the structure of Hearn and Demaine's hardness proof of NCL[7] we may assume that such an embedding of G is given, but otherwise we can compute one in $O(n)$ time, for example with an algorithm of Biedl and Kant.[9] The next step is to replace each vertex and edge by appropriate *gadgets* built out of points that are to receive labels. In the figure, our gadgets are depicted in the circular insets. We call the points that have been depicted with dark gray labels *blockers*, as we can consider their labels to be fixed obstacles: labeling them differently than shown will only restrict the placement of other labels further. As is, the depicted blockers restrict the red and blue labels to two possible positions each in the 4-position model. These correspond to the two different orientations of edges. Labels "directed" *into* a vertex gadget (such as the red labels in the depicted AND gadget) correspond to edges pointing *out* of the vertex. Conversely, labels directed *out* of a vertex gadget (such as the blue labels in the depicted AND gadget) correspond to edges pointing *into* the vertex. The inflow constraints are then enforced by the light gray labels, which restrict the amount of usable space for the red and blue labels inside the vertex gadget.

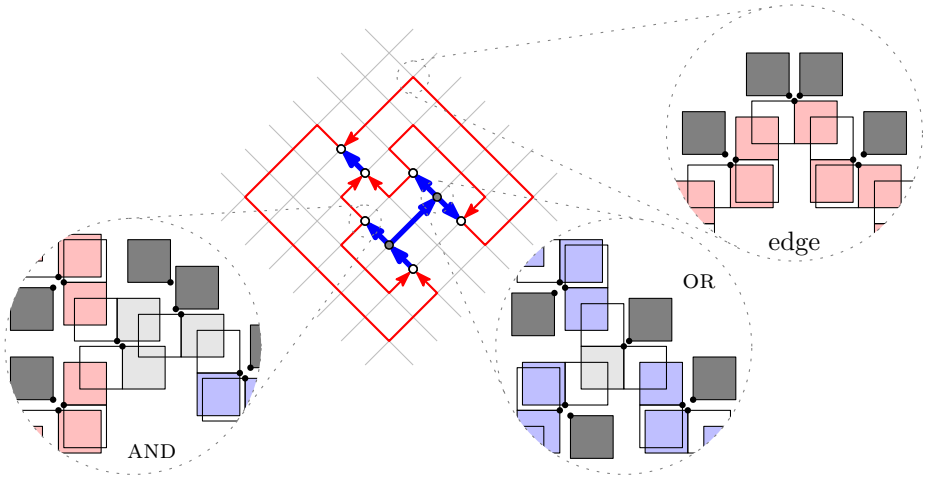

Fig. 3. Our construction for simulating non-deterministic constraint logic with dynamic point labeling.

2.2. *Gadgets*

Figures 4–6 show our vertex and edge gadgets in more detail. As mentioned before, the blockers restrict the colored labels marked A, B, and C to two possible positions each in the 4-position model. We call the label position closest to the center of the vertex gadget *inward*, and the other *outward*. In the figures, labels A and B are placed inward, and label C is placed outward. In the slider models, labels can take on any position in between these two extremes, but we may assume that only a very small range of positions is actually used. Consider, for example, label A in Figure 4. Without moving label A', we can only move A left by at most ε, or down by at most 2ε. We refer to positions for A from this range as *inward*, and define the term similarly for B and C. If (and only if) A' is moved down by at least $1 - \varepsilon$, then A can move further leftward. We may then move A all the way to its leftmost position, and there is no reason not to do so. Thus we may define *outward* as in the 4-position model.

With this terminology in place, we shall prove that our construction faithfully simulates a constraint graph, with labels placed *inward* corresponding to edges directed *out* of a vertex, and labels placed *outward* corresponding to edges directed *into* a vertex. For this we will have to show that any overlap-free labeling of the depicted points corresponds to a legal configuration of the constraint graph, and that all legal configurations can be represented in this way. Furthermore, we need to show that a move can be made in the constraint graph if and only if the corresponding label movement can be done continuously without overlap. We shall first show this is the case locally for the vertex gadgets, and then show how our edge gadgets make

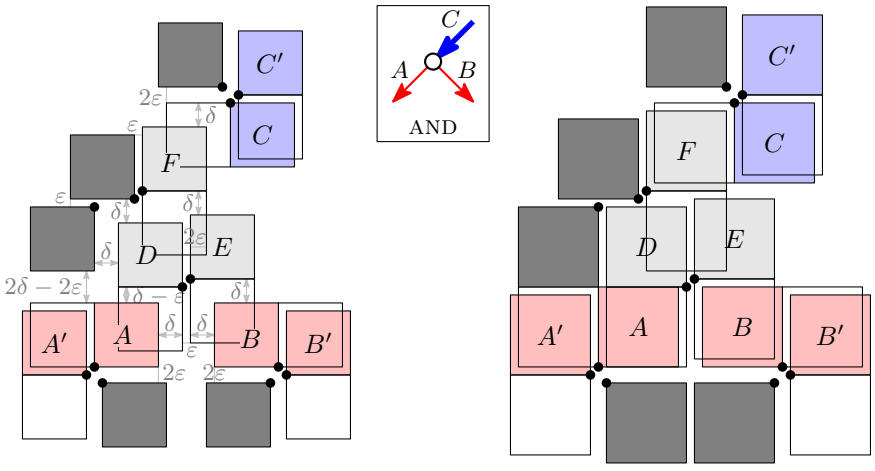

Fig. 4. Our gadget simulating the AND vertex shown in the inset at the top. The construction works for square labels of side length 1 (left) up to $1 + \delta - \varepsilon$ (right). In the figure, $\delta = 3/8$ and $\varepsilon = 1/8$.

this true globally.

Lemma 1. *The construction in Figure 4 satisfies the same constraints as an NCL* AND *vertex with incident edges A, B, and C, where A and B are the edges of weight* 1*. This holds for the 4-position model when* $0 < \varepsilon \leqslant \delta < 1 - 3\varepsilon$*, and for the 2-slider and 4-slider models when* $0 < \varepsilon \leqslant \delta < 1/3 - 4\varepsilon/3$*.*

Proof. As argued above we may assume the blockers are always labeled as depicted. In a labeling without overlap the labels marked A, B, and C can then only be placed inward or outward, corresponding to the opposite orientation of the respective incident edges of the AND vertex. To show that the vertex's inflow constraint is always satisfied, we must show that in an overlap-free labeling either C must be placed outward, or both A and B must be placed outward. For the 4-position model this is fairly easy to see. The depicted situation has A and B placed inward and C placed outward. Placing C inward can be done if and only if we place D, E, and F downward, which in turn necessitates placing A and B outward. For the 2-slider and 4-slider models we will show the same by analyzing the gap between F and the blocker nearest to C. To maximize this gap, first move A and B down by 2ε so they touch their blockers. With unit-square labels, D and E can then move down by $\delta + \varepsilon$ and $\delta + 2\varepsilon$, respectively. Finally, F may move down by $2\delta + 2\varepsilon$. The resulting gap above F then has size $3\delta + 4\varepsilon$, and cannot be widened any further. Since $3\delta + 4\varepsilon < 3(1/3 - 4\varepsilon/3) + 4\varepsilon = 1$, label C does not fit in this gap, meaning it cannot be placed inward. Increasing the label size will make the gap even narrower.

It remains to argue that labels can be moved if and only if the corresponding edge reversals are possible in the constraint graph. It follows from the preceding paragraph that we cannot perform any label moves not corresponding to legal moves on the constraint graph. Thus we need to show only that all legal moves can be performed as label motions. Such label motions are easily produced. We simply move D downward when A moves outward, and move D upward when A moves inward. Similarly, E should move along with B, and F with C. □

Lemma 2. *The construction in Figure 5 satisfies the same constraints as an NCL protected* OR *vertex with incident edges A, B, and C, where A and B form the protected edge pair. This holds for the 4-position model when* $0 < \varepsilon \leqslant \delta < 1 - 3\varepsilon$*, and for the 2-slider and 4-slider models when* $0 < \varepsilon \leqslant \delta < 1/3 - 4\varepsilon/3$*.*

Proof. Recall that in a protected OR vertex, the protected edges can be assumed to never both be directed into the vertex, because of the inflow constraints of neighboring AND vertices. We may therefore assume in our protected OR gadget that the labels A and B are never both placed outward. We now need to show that it is not possible to place A, B, and C all inward simultaneously. Suppose we would place all three inward. In the 4-position model, clearly at least one of them must overlap with D. To see this for the 2-slider and 4-slider models, recall that *inward* means

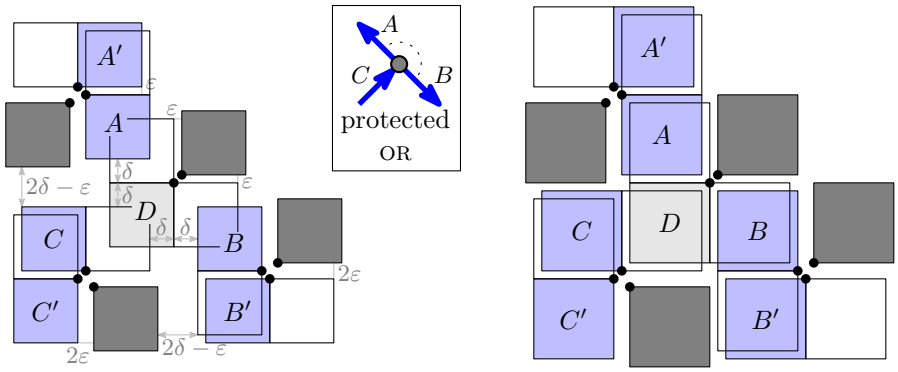

Fig. 5. Our gadget simulating the protected OR vertex shown in the inset at the top. The construction works for square labels of side length 1 (left) up to $1+\delta-\varepsilon$ (right). In the figure, $\delta = 3/8$ and $\varepsilon = 1/8$.

that A, B, and C have all been displaced by at least $1-\varepsilon$ from their most *outward* position. With unit-square labels this leaves a gap of $2\delta+\varepsilon$ between A and C, which can be increased to $2\delta+3\varepsilon$ by moving C towards its blocker, but cannot be widened any further. Since $2\delta + 3\varepsilon < 2(1/3 - 4\varepsilon/3) + 3\varepsilon = 2/3 + \varepsilon/3 < 1$, label D does not fit in this gap. Similar reasoning applies to the gap between B and C. Increasing the label size will make the gaps even narrower.

We next need to show that labels can be moved if and only if the corresponding edges can be reversed. The preceding paragraph shows that we cannot simulate illegal moves with label motions. It remains to show that all legal moves can be simulated with label motions. Moving one of the labels A, B, or C outward is always possible. To move one of them inward is only problematic if label D is currently in the way. We then need to move D out of the way first. When moving C inward this is easy, as D can always go either to the A side or to the B side. When moving A or B inward a problem would occur if C was already placed inward. Then D would have to "jump" over C. Note, however, that this requires both A and B to be placed outward prior to moving one of them inward. Such a scenario cannot occur because the OR vertex is protected. □

Having demonstrated working vertex gadgets, it remains to show that we can connect them over longer distances in such a way that a label moving out of one vertex gadget causes a label of another vertex gadget to move inward, and vice versa. This is done by building edge gadgets out of the pieces on the left in Figure 6. By interleaving *bend pieces* with chains of *wire pieces*, and varying the spacing between wire pieces appropriately, any desired polygonal chain of diagonal segments can be simulated. An example is shown on the right in the figure. By the same process an edge gadget can be connected to its two vertex gadgets, as the three colored ends of each vertex gadget are simply wire pieces. As an example, we may identify the

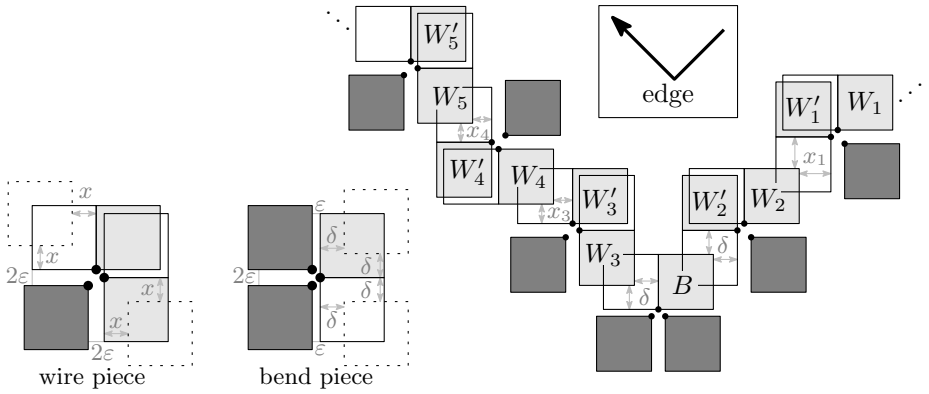

Fig. 6. (Left) The building pieces for our edge gadgets. They may be rotated arbitrarily in 90° increments, and "connected" together along the dotted lines. (Right) An example gadget built with these pieces, simulating the edge shown in the inset at the top. The construction is the same for (red) weight-1 edges and (blue) weight-2 edges, and works for square labels of side length 1 up to $1 + \delta - \varepsilon$. In the figure, $\delta = 3/8$ and $\varepsilon = 1/8$. The distance marked x may be varied along an edge gadget in order to make it line up correctly with vertex gadgets. For the 4-position model, $x \in [\delta - \varepsilon, 1)$; for the 2-slider and 4-slider models, $x \in [\delta - \varepsilon, 1 - 3\varepsilon)$.

points labeled W_1 and W'_1 on the right in Figure 6 with the points labeled by A and A' (respectively) in Figure 4.

Lemma 3. *A label on one end of an edge gadget (such as the one in Figure 6) may only move out of its vertex gadget if the label on the other end is placed into its vertex gadget. This holds for the 4-position model when* $0 < \varepsilon \leqslant \delta < 1 - 3\varepsilon$*, and for the 2-slider and 4-slider models when* $0 < \varepsilon \leqslant \delta < 1/3 - 4\varepsilon/3$.

Proof. In the figure, W_1 is placed inward relative to a non-pictured vertex gadget above and to its right. Suppose we want to move it outward. This means W_1 moves left by more than ε, which can only happen if W'_1 moves downward by at least $1 - \varepsilon$. Moving W_2 down by 2ε so that it touches its blocker is not sufficient to accommodate this, because $x_1 + 2\varepsilon < (1 - 3\varepsilon) + 2\varepsilon = 1 - \varepsilon$. The only option is therefore to move W_2 left by at least $1 - x_1 > \varepsilon$, for which W'_2 has to move down by at least $1 - \varepsilon$. In turn, B then has to move left by at least $1 - \delta > \delta + 2\varepsilon$, for which W_3 has to move up by $1 - \delta > \varepsilon$, for which W'_3 has to move left by $1 - \varepsilon$, and so forth. Symmetric reasoning can be used to show that such a "signal" can also traverse the edge gadget in the opposite direction (from W'_5 to W_1). □

Lemma 3 implies that an edge gadget cannot have both ends placed outward of their respective vertex gadgets. On the other hand, both ends *can* be placed inward. This corresponds to an edge of the constraint graph being directed into *neither* of its two incident vertices. To obtain a correspondence between configurations of a constraint graph and labelings of our point set we therefore modify the definition of

a legal configuration to allow such "unoriented" edges. A move then either reverses an oriented edge, unorients an oriented edge, or orients an unoriented edge. This does not meaningfully change NCL: if all inflow constraints are satisfied with some unoriented edges, they remain satisfied when such edges are oriented arbitrarily.

2.3. *Putting it all together*

The results of this section now lead to the following theorem.

Theorem 1. *Let G be a planar constraint graph with n vertices, and let ε and δ be two real numbers with $0 < \varepsilon \leqslant \delta < 1 - 3\varepsilon$. One can then construct a point set $P = P(G, \delta, \varepsilon)$ in polynomial time that has the following properties for any $s \in [1, 1 + \delta - \varepsilon]$:*

- *the cardinality of P is polynomially bounded in n,*
- *the x- and y-coordinates of the points in P are linear combinations of 1, δ, and ε, with integer coefficients that are polynomially bounded in n,*
- *for any legal configuration of G there is an overlap-free static labeling of P with $s \times s$ square labels in the 4-position model and vice versa,*
- *there is a sequence of moves transforming one legal configuration of G into another if and only if there is a dynamic labeling transforming the corresponding static labelings of P into each other.*

When $\delta < 1/3 - 4\varepsilon/3$, the same results hold for the 2- and 4-slider models.

Proof. As mentioned above, the given constraint graph G may be assumed to be embedded on a grid, with its vertices on grid vertices and its edges being interior-disjoint paths along grid lines. Such a structure follows from Hearn and Demaine's hardness proof of NCL,[7] but we could otherwise achieve it in $O(n)$ time, for example with an algorithm of Biedl and Kant.[9] We turn this grid by 45° relative to the coordinate axes, and replace G's vertices and edges by the appropriate gadgets of Lemma 1 through 3, as was illustrated in Figure 3. If we choose the scale of the grid and the spacing between the wire pieces of edge gadgets appropriately, then the gadgets will fit together perfectly. The union of the points making up the gadgets forms the desired set P.

The third and fourth claim of the theorem now follow from Lemma 1 through 3. To see that the first two claims hold, note that the coordinates of all points within our gadgets are linear combinations of 1, δ, and ε, with integer coefficients. Furthermore, the amount of freedom in the alignment of wire pieces allows us to maintain this property when connecting gadgets together. Thus we only need to show that a polynomial number of wire pieces is used, so that the number of points and these integer coefficients cannot become super-polynomial. This follows immediately when using Biedl and Kant's algorithm, as they lay out G on an $n \times n$ grid with at most $2n + 2$ edge bends in total. □

2.4. *Hardness of static point labeling revisited*

In Section 3 we will use the construction described above to prove the PSPACE-hardness of dynamic point labeling problems. The construction can also be used, however, to prove NP-hardness of static point labeling problems. For the 4-position and 4-slider models this simply yields a new proof of known results, but for the 2-slider model we are not aware of any previously published proof.

Corollary 1. *The following decision problem is strongly NP-complete for the 4-position, 2-slider, and 4-slider label models.*

> ***Given:*** *A point set P.*
> ***Decide:*** *Whether there exists a static labeling $\mathcal{L}$ for P which labels all points with non-overlapping unit-square labels.*

Additionally, unless $P = NP$, the maximum label size for which there is such a static labeling $\mathcal{L}$ cannot be $(4/3-\varepsilon')$-approximated in polynomial time for any $\varepsilon' > 0$. For the 4-position model, it cannot even be $(2-\varepsilon')$-approximated.

Proof. In Theorem 5.4 of their book on constraint logic,[8] Hearn and Demaine show that it is NP-hard to decide whether a given constraint graph G has a legal configuration. By Theorem 1 above we can construct a point set $P = P(G, \delta, \varepsilon)$ that can be labeled without overlap if and only if G has a legal configuration. Thus the labeling problem is NP-hard. Moreover, the size of P and its coordinates is polynomially bounded in the size of G, making the problem strongly NP-hard. The construction works for identical square labels with a side length in the range $[1, 1+\delta-\varepsilon]$. We must have $0 < \varepsilon \leqslant \delta < 1-3\varepsilon$ for the 4-position model, so pick $\varepsilon = \varepsilon'/4$ and $\delta = 1-\varepsilon'$ to obtain the claimed hardness-of-approximation result. For the 2-slider and 4-slider models $\delta < 1/3 - 4\varepsilon/3$ must hold, so pick $\delta = 1/3 - \varepsilon'$ instead. Membership in NP was shown previously for the 4-position[2] and 4-slider[4] models, and the latter proof can easily be modified to show this for the 2-slider model as well. □

For deciding if all points can be labeled without overlap we now know exactly where the boundary between easy and hard label models lies. The task is in P for the 1-position model (a rectangle intersection problem) and the 2-position model (by reduction to 2-SAT[2]), but becomes strongly NP-complete for the 3-position[3] and 4-position[2] models (and their supersets). For the 1-slider model a greedy left-to-right labeling[4] will work, but the problem is strongly NP-complete for the 2-slider model and the 4-slider model.[4]

In addition, we now have a single, unified framework to derive all of these NP-hardness results, with the single exception of the 3-position model. Perhaps it is possible to modify our construction to encompass this case as well. We shall not attempt to do so now, however, and will instead focus on applying the framework to the complexity of *dynamic* point labeling.

3. Hardness of dynamic point labeling

In this section we will define a number of dynamic point-labeling problems. The input to all of them is a dynamic point set P, which specifies for each point $p \in P$ an arrival and departure time, as well as a continuous trajectory. In some problems these changes to the point set may be fairly arbitrary, in others they must be the result of the user panning, rotating, or zooming their viewport of a static point set. In all cases we seek a dynamic labeling $\mathcal{L}$ of P for a given time interval $[a, b]$. That is, $\mathcal{L}(t)$ must be a static labeling of $P(t)$ for all $t \in [a, b]$, and each of $\mathcal{L}$'s labels must move continuously over time.

Various optimization questions may be formulated for dynamic labeling. We may disallow label overlap entirely, and then seek a dynamic labeling that labels as many points as possible for as long as possible. Alternatively, perhaps we wish to label all points at all times, and wish to have as little label overlap as possible. Whatever the case may be, labeling all points at all times without any overlap is likely the most desirable outcome. Unfortunately, we will see that deciding whether such a solution exists is already strongly PSPACE-complete, even if we drastically restrict the dynamic nature of the point set. Thus, all optimization problems of this kind are strongly PSPACE-hard.

3.1. *Transforming one labeling into another*

As a warm-up, we start with a very simple PSPACE-hardness proof. The first problem we consider does not concern a dynamic point set or viewport at all. Instead, we are given two distinct static labelings $\mathcal{L}(a)$ and $\mathcal{L}(b)$ for one static point set P. We then want to find a dynamic labeling $\mathcal{L}$ that transforms $\mathcal{L}(a)$ into $\mathcal{L}(b)$ over the time interval $[a, b]$ by continuous label movement, without any labels ever overlapping.

Theorem 2. *The following decision problem is strongly PSPACE-hard for the 4-position, 2-slider, and 4-slider label models.*

> ***Given:*** *A static point set P, numbers a and b with $a < b$, and two static labelings $\mathcal{L}(a)$ and $\mathcal{L}(b)$ of P using non-overlapping unit-square labels.*
> ***Decide:*** *Whether there exists a dynamic labeling $\mathcal{L}$ for P with $\mathcal{L}(a)$ and $\mathcal{L}(b)$ as given, which labels all points with non-overlapping unit-square labels over the time interval $[a, b]$.*

Additionally, unless P = PSPACE, the maximum label size for which there is such a dynamic labeling $\mathcal{L}$ cannot be $(4/3 - \varepsilon')$-approximated in polynomial time for any $\varepsilon' > 0$. For the 4-position model, it cannot even be $(2 - \varepsilon')$-approximated.

Proof. To show PSPACE-hardness, we reduce from configuration-to-configuration NCL. Thus, we are given a constraint graph G with two legal configurations $\mathcal{C}_A$ and $\mathcal{C}_B$ of its edges, and want to decide whether there is a sequence of moves transform-

ing $\mathcal{C}_A$ into $\mathcal{C}_B$. The reduction is much the same as in Corollary 1. By Theorem 1, we can construct a point set $P = P(G, \delta, \varepsilon)$ with two valid static labelings $\mathcal{L}(a)$ and $\mathcal{L}(b)$ corresponding to configurations $\mathcal{C}_A$ and $\mathcal{C}_B$. The desired dynamic labeling then exists if and only if there is a sequence of moves transforming $\mathcal{C}_A$ into $\mathcal{C}_B$, and deciding the latter is PSPACE-hard. All coordinates are polynomially bounded in the size of G, making the decision problem strongly PSPACE-hard. The construction works for identical square labels with a side length in the range $[1, 1 + \delta - \varepsilon]$. We must have $0 < \varepsilon \leqslant \delta < 1 - 3\varepsilon$ for the 4-position model, so pick $\varepsilon = \varepsilon'/4$ and $\delta = 1 - \varepsilon'$ to obtain the claimed hardness-of-approximation result. For the 2-slider and 4-slider models $\delta < 1/3 - 4\varepsilon/3$ must hold, so pick $\delta = 1/3 - \varepsilon'$ instead. □

3.2. *Labeling dynamic point sets*

Now suppose only one of $\mathcal{L}(a)$ and $\mathcal{L}(b)$ is given, and we are free to choose the other; does there exist a dynamic labeling then? For a static point set the answer is trivially "yes": simply set $\mathcal{L}(a) = \mathcal{L}(b) = \mathcal{L}(t)$ for all $t \in [a, b]$. But when labeling a dynamic point set where points move, or in which points are added and removed, this question becomes hard.

Theorem 3. *The following decision problem is strongly PSPACE-hard for the 4-position, 2-slider, and 4-slider label models.*

> ***Given:*** *A dynamic point set P (with given trajectories, arrival times, and departure times), numbers a and b with $a < b$, and a static labeling $\mathcal{L}(a)$ for $P(a)$.*
> ***Decide:*** *Whether there exists a dynamic labeling $\mathcal{L}$ for P respecting $\mathcal{L}(a)$ that labels all points with non-overlapping unit-square labels over the time interval $[a, b]$.*

Additionally, unless $P = PSPACE$, the maximum label size for which there is such a dynamic labeling $\mathcal{L}$ cannot be $(4/3 - \varepsilon')$-approximated in polynomial time for any $\varepsilon' > 0$. For the 4-position model, it cannot even be $(2 - \varepsilon')$-approximated.

All of the above remains true when

- *all points are stationary, and during $[a, b]$ one point is removed and one point is added,*
- *no points are added or removed, and all points move at the same, constant speed along parallel (but possibly opposite), straight-line trajectories, or*
- *no points are added or removed, and all points but one are stationary.*

Proof. To show PSPACE-hardness when $\mathcal{L}(a)$ is given, we reduce from configuration-to-edge NCL. Thus we are given a constraint graph G with a legal starting configuration $\mathcal{C}_A$ and the goal orientation of a single edge e_B, and want to decide whether there is a sequence of moves on G starting from $\mathcal{C}_A$ that will result in e_B having its specified orientation. By Theorem 1, we may construct a point set $P = P(G, \delta, \varepsilon)$ and a valid static labeling $\mathcal{L}(a)$ corresponding to configuration $\mathcal{C}_A$. Now pick one blocker $q \in P$ in the edge gadget for e_B, and suppose $p \in P$ is the

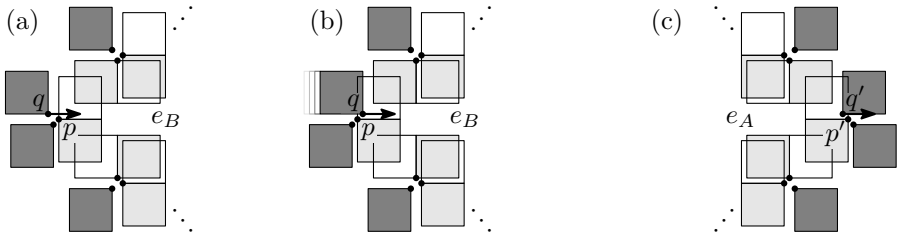

Fig. 7. Our reduction from edge-to-edge NCL to dynamic labeling of moving points. (a) The modified gadget for edge e_B at time a. (b) The modified gadget for edge e_B at time b. (c) The modified gadget for edge e_A at time a.

point for which q blocks some label candidates. We now make q move at a constant speed of $v = 2\varepsilon/(b-a)$ towards the nearest non-blocked candidate of p, as in Figure 7(a). Figure 7(b) shows the end result at time b: the edge gadget has become constrained to a single orientation around p. Thus there is a sequence of moves on $\mathcal{C}_A$ resulting in e_B having the specified orientation if and only if there is a dynamic labeling $\mathcal{L}$ for this dynamic point set starting at $\mathcal{L}(a)$. The same result may be obtained by moving q at speed $v/2$ and moving all other points at speed $v/2$ in the opposite direction. Alternatively, we may remove the point q from P and reinsert it at its modified location. In conclusion, all the stated problem formulations are PSPACE-hard.

To prove the remaining claims (strong hardness of the decision problem, and hardness of approximating the maximum label size) proceed in the same way as in Theorem 2. □

By symmetry, the hardness claims in Theorem 3 remain true when $\mathcal{L}(b)$ is given instead of $\mathcal{L}(a)$, as one can simply exchange the roles of a and b in the proof. When neither $\mathcal{L}(a)$ nor $\mathcal{L}(b)$ is given, so that both can be chosen freely, the problem does not become much easier either.

Theorem 4. *The following decision problem is strongly PSPACE-hard for the 4-position, 2-slider, and 4-slider label models.*

Given: *A dynamic point set P (with given trajectories, arrival times, and departure times), and numbers a and b with $a < b$.*
Decide: *Whether there exists a dynamic labeling $\mathcal{L}$ for P that labels all points with non-overlapping unit-square labels over the time interval $[a, b]$.*

Additionally, unless P = PSPACE, the maximum label size for which there is such a dynamic labeling $\mathcal{L}$ cannot be $(4/3-\varepsilon')$-approximated in polynomial time for any $\varepsilon' > 0$. For the 4-position model, it cannot even be $(2-\varepsilon')$-approximated.

All of the above remains true when

- *all points are stationary, and during $[a, b]$ two points are removed and two points*

are added,

- *no points are added or removed, and all points move at the same, constant speed along parallel (but possibly opposite), straight-line trajectories, or*
- *no points are added or removed, and all but two points are stationary.*

Proof. To show PSPACE-hardness, we reduce from edge-to-edge NCL. Thus we are given a constraint graph G with orientations for two edges e_A and e_B, and want to decide whether there is a sequence of moves on G starting from a legal configuration $\mathcal{C}_A$ where e_A has its specified orientation and ending in a legal configuration $\mathcal{C}_B$ where e_B has its specified orientation. We proceed in the same way as in Theorem 3, but now modify blockers in *two* edge gadgets. For e_B we make the blocker q move into the edge gadget at speed v as before, for e_A we start the blocker q' inside the edge gadget and make it move outward at speed v, as in Figure 7(c). We pick a slightly faster speed $v = 3\varepsilon/(b-a)$, so that e_A is constrained only during the time interval $[a, a+\Delta)$, and e_B only during the time interval $(b-\Delta, b]$, where $\Delta = (b-a)/3$. If we take care in how the edge gadgets are laid out on the grid, then we can always ensure that the two blockers move in the same direction (as they do in the figure). We may then reduce their speeds to $v/2$, and move all other points at speed $v/2$ in the opposite direction. Alternatively, at time $a+\Delta$ we may remove the one blocker and re-insert it at its new location, and do the same for the other blocker at time $b-\Delta$. The remaining claims can be proved as in preceding theorems. □

3.3. *Labeling under panning, zooming, and/or rotation*

In interactive mapping applications, users are presented with a rectangular *viewport* V showing a portion of a larger map. By dynamically panning, rotating, and/or zooming the map, the user controls which portion of the map is displayed at any given time. The task of labeling the points inside V can be seen as a special case of labeling dynamic point sets. Continuous panning, rotation, and zooming of the map may cause points to enter or leave V at its boundary, and causes all points within V to move on continuous trajectories. We will require only the points inside V to be labeled, and these labels must be fully contained in V. Points outside of V need not be labeled, but we may wish to do so in order to ensure a smooth dynamic labeling. Otherwise a label would have to instantly appear or disappear whenever its point hits the boundary of V. Regardless of whether we allow such "popping" of labels, it turns out that the question of whether a dynamic labeling can label all points in the viewport without overlap is strongly PSPACE-hard. We prove this first for panning.

Theorem 5. *The following decision problem is strongly PSPACE-hard for the 4-position, 2-slider, and 4-slider label models.*

Given: *A closed rectangle V panning along a given trajectory, a set of points P, and two numbers a and b with $a < b$.*
Decide: *Whether there exists a dynamic labeling $\mathcal{L}$ for P that labels $P \cap V(t)$ with non-overlapping unit-square labels inside $V(t)$ for all $t \in [a, b]$.*

This is true regardless of

- *whether panning occurs on a straight line at constant speed or on some other trajectory,*
- *whether points on the boundary of V may instantly lose/gain their labels or not, and*
- *whether the labelings $\mathcal{L}(a)$ and/or $\mathcal{L}(b)$ are given or not.*

Proof. We prove PSPACE-hardness for the 4-slider model, with neither $\mathcal{L}(a)$ nor $\mathcal{L}(b)$ being given. The remaining results can then be derived by similar techniques as in previous theorems. Our reduction is from edge-to-edge NCL, where we are given a constraint graph G and orientations for two of its edges e_A and e_B. We construct the point set $P = P(G, \delta, \varepsilon)$ as usual, but this time we lay it out on the grid in such a way that e_A is on the far left and e_B is on the far right, as in Figure 8. While this is always possible, it may introduce edge crossings. However, Hearn and Demaine show[7] how to construct a planar "cross-over" in NCL that functions as two crossing edges. This cross-over uses only AND and protected OR vertices, so we may emulate it using our gadgets.

Next, we add two additional points q and q', with q to the left of e_A by $1 - 2\varepsilon$ and q' to the right of e_B by $1-3\varepsilon$, as in Figure 8. We construct a viewport rectangle V with its left side containing q, and its right side ε to the left of q'. This forces us to label q in a way that constrains e_A to a single orientation. Now suppose the points move to the left (that is, the view pans to the right). This lifts this constraint on e_A, as we may then move q's label out of V (either immediately, or over a short time interval). Panning may continue for a distance of $1 - \varepsilon$ before disturbing the functioning of the edge gadget for e_A. However, already after moving by ε are we

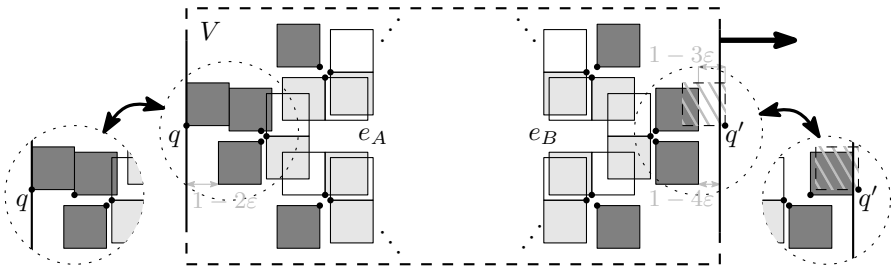

Fig. 8. Our reduction from edge-to-edge NCL to dynamic labeling under panning. The main figure shows the construction for the 4-slider model, the circular insets at the left and right show how to modify the construction for the 4-position model.

required to start labeling point q' in a way that constrains e_B to a single orientation. Thus there exists a dynamic labeling of P for this panning motion if and only if there exists a sequence of moves in G starting with e_A in its specified orientation and ending with e_B in its specified orientation. This makes the problem PSPACE-hard for the 4-slider model. For a horizontal 2-slider model we may use the exact same construction; for a vertical 2-slider model we simply rotate the construction by 90°, placing e_A and e_B at the top and bottom. For the 4-position model, the construction needs a small modification so that labels always contain their point on a corner. This modification is depicted in the circular insets of Figure 8.

The cases that either $\mathcal{L}(a)$ or $\mathcal{L}(b)$ is given are similar, using a reduction from configuration-to-edge NCL that modifies only one edge gadget. If both $\mathcal{L}(a)$ and $\mathcal{L}(b)$ are specified, the problem is already hard without any point ever touching the viewport, as we saw in Theorem 2. Strong PSPACE-hardness may be proved as in Theorem 2. □

When panning, all points move in the same direction along parallel trajectories at identical speeds. When rotating, however, the points move on concentric circles centered on the point c at the center of the viewport V. The speed at which a point p moves is proportional to the distance $|p - c|$. Our hardness reduction for panning can be modified for the case of rotation as follows.

Theorem 6. *The following decision problem is strongly PSPACE-hard for the 4-position, 2-slider, and 4-slider label models.*

> ***Given:*** *A closed rectangle V with center c, a set of points P rotating around c, and two numbers a and b with $a < b$.*
> ***Decide:*** *Whether there exists a dynamic labeling $\mathcal{L}$ for P that labels $P(t) \cap V$ with non-overlapping unit-square labels inside V for all $t \in [a, b]$.*

This is true regardless of

- *whether the point set is rotated back and forth arbitrarily, or only in a single direction at constant speed,*
- *whether points on the boundary of V may instantly lose/gain their labels or not, and*
- *whether the labelings $\mathcal{L}(a)$ and/or $\mathcal{L}(b)$ are given or not.*

Proof. Consider again the reduction from NCL to dynamic labeling under panning that was depicted in Figure 8. Now suppose V has been constructed so that q and q' both lie below its center c. Rotating P clockwise around c then moves q out of V, and q' towards V. This also changes the horizontal and vertical distances between the points of the gadgets. As long as these changes are less than ε, however, all gadgets will still work. Thus, a sufficiently small rotation will not disturb our reduction. To ensure q' ends up inside V during this rotation, decrease the initial distance between V and q' appropriately by moving the right side of V slightly farther to

the right. With these observations we may now prove all claims similarly as in Theorem 5. □

When zooming, each point $p \in P$ moves on the ray from c through p, at a speed proportional to the distance $|p - c|$. When zooming out the points move towards c, when zooming in the points move away from c. This adds an asymmetry to the problem not present for panning or rotation: it now matters *which* static labeling is given. Suppose during $[a, b]$ the view is maximally zoomed out at time $t \in [a, b]$, and that it is the labeling $\mathcal{L}(t)$ for $P(t)$ that is given. Then the answer to the decision problem is trivially "yes", since zooming in further cannot "invalidate" this given labeling: all inter-point distances increase and no new points can enter V. If we are not given any static labelings to adhere to, then the problem is "merely" NP-complete. We simply determine the time $t \in [a, b]$ at which the view is maximally zoomed out, and then solve the corresponding static labeling problem on $P(t)$. But when given a single static labeling, and not the one where the view is maximally zoomed out, then the problem is PSPACE-hard. (Of course when both $\mathcal{L}(a)$ and $\mathcal{L}(b)$ are given the problem is also PSPACE-hard, as this was already proven in Theorem 2.)

Theorem 7. *The following decision problem is strongly PSPACE-hard for the 4-position, 2-slider, and 4-slider label models.*

> ***Given:*** *A closed rectangle V with center c, a set of points P scaling around c, numbers a and b with $a < b$, and a static labeling $\mathcal{L}(c)$ of $P(c)$ with $c \in \{a, b\}$. Here c must not be the time at which P's scaling factor is minimal during $[a, b]$.*
> ***Decide:*** *Whether there exists a dynamic labeling $\mathcal{L}$ for P that labels $P(t) \cap V$ with non-overlapping unit-square labels inside V for all $t \in [a, b]$.*

This is true regardless of

- *whether the point set is scaled back and forth arbitrarily, or only in a single direction at constant speed, and*
- *whether points on the boundary of V may instantly lose/gain their labels or not.*

Proof. Consider yet again the reduction from NCL to dynamic labeling under panning that was depicted in Figure 8. Suppose we remove the point q' from the construction, and then start zooming in. This will cause q to move out of V, and the constraint on e_A will be lifted. Conversely, suppose we remove point q from the construction, and then start zooming out. This will cause q' to move into V, adding a constraint on e_B. Thus we can reduce configuration-to-edge NCL to dynamic labeling under zooming. As with rotation, we will have to be careful that the changes to inter-point distances do not disturb the gadgets. The solution is the same as it was then: move the right side of V closer to q' so that we do not have to zoom so far as to disturb the gadgets. □

3.4. *Membership in PSPACE*

We have shown that even very restricted variants of dynamic point labeling are PSPACE-hard. Now, we shall show that a very general version of dynamic point labeling is in PSPACE. Here points may arrive and depart at will over the interval $[a, b]$, they may move on arbitrary algebraic trajectories of bounded degree, and their labels can be rectangles of arbitrary sizes that need not be identical. Since this general variant encompasses all previously mentioned variants, this implies that all of them are PSPACE-complete.

Theorem 8. *The following decision problem is in PSPACE for the 4-position, 2-slider, and 4-slider label models.*

> ***Given:*** *A dynamic point set P (with given arrival times, departure times, bounded-degree algebraic trajectories, and label dimensions) and numbers a and b with $a < b$.*
> ***Decide:*** *Whether there exists a dynamic labeling $\mathcal{L}$ for P that labels all points with non-overlapping axis-aligned rectangles of the specified dimensions over the time interval $[a, b]$.*

The above is true regardless of whether the labelings $\mathcal{L}(a)$ and/or $\mathcal{L}(b)$ are given or not.

Proof. We shall describe a non-deterministic algorithm running in polynomial space, showing that the problem is in NPSPACE. Since PSPACE=NPSPACE by Savitch's theorem,[10] this then yields the claimed result. We shall show this for the 4-slider model first, while assuming that $\mathcal{L}(a)$ and $\mathcal{L}(b)$ are *not* given, and that no points arrive or depart between times a and b. The minor modifications needed for the other cases and label models will be explained at the end of the proof.

A key idea for our algorithm is the notion of a *canonical (static) labeling*, defined as follows. Draw vertical lines through the left and right sides of all labels. We say a label is *canonical along the x-axis* if it is placed entirely to the left or right of its point, or if its left or right side lies on a vertical line defined by another label which is canonical along the x-axis. We define a label to be *canonical along the y-axis* in an analogous fashion. A label is called canonical if its position is canonical along both axes. A static labeling is called canonical if all of its labels are canonical (see Figure 9).

A canonical static labeling can be represented combinatorially by storing a pair of symbolic coordinates for each label. With n labels, there are at most $2+4(n-1) = 4n-2$ possible x-positions for each label: to the left or to the right of its point, or with its left or right side aligned with the left or right side of another label. Similarly, there are at most $4n-2$ possible y-positions. Our algorithm starts out by non-deterministically guessing such a combinatorial structure for a canonical static labeling $\mathcal{L}(t_1)$, where $t_1 = a$. Then, it computes the time $t_2 > t_1$ of the first *event*. That is, it computes the first time that two horizontal lines or two vertical

Fig. 9. (a) An example of a canonical labeling. (b) Something that is *not* a canonical labeling: in a sequence of labels whose sides are aligned, at least one must be in an extreme position along that axis.

lines defined by the labels cross each other, assuming the labels maintain fixed positions relative to the points they label. Until time t_2 it is definitely possible to maintain the label positions specified by $\mathcal{L}(t_1)$, but after time t_2 a new set of label positions may be needed to avoid overlap. To this end the algorithm also guesses a sequence of zero or more *label moves*. These are to be applied in sequence to the labeling $\mathcal{L}(t_1)$, resulting in a new labeling $\mathcal{L}(t_2)$. Each label move specifies a label and a direction (either clockwise or counter-clockwise), and applying it to a labeling means moving the label in the specified direction until it reaches a new canonical position. A label move does not have to specify a speed for the label, or a subinterval of (t_1, t_2) at which it occurs. For the combinatorial structure of the labeling it makes no difference if we assume that all label moves happen instantly one after another at unspecified times between t_1 and t_2. Having applied the label moves, the combinatorial structure of a new canonical labeling $\mathcal{L}(t_2)$ results, for which the next event time $t_3 > t_2$ is computed, and a new sequence of label moves is guessed. We continue in this fashion until reaching an event time $t_m \geqslant b$ for some m. The result is a sequence of canonical labelings at event times, interspersed with sequences of label moves. We know this sequence is finite, because: 1) the event times correspond to distinct zeroes of a polynomial that is a function of the point trajectories, and 2) any canonical labeling can be reached from any other in a finite number of label moves. Such a sequence will be referred to as a *canonical dynamic labeling*, and we claim that it exists if and only if there also exists a proper dynamic labeling.

To prove the claim in the "only if" direction, suppose that $\mathcal{L}$ is a canonical dynamic labeling for P. To make $\mathcal{L}$ a proper dynamic labeling, all that is needed is to assign disjoint and non-empty intervals of time to all its label moves. As the event times are distinct, this can always be done, for example by dividing the interval between two events evenly among its label moves.

To prove the claim in the "if" direction, suppose there is a dynamic labeling $\mathcal{L}$ for P over the interval $[a, b]$. Let $t_1 = a$ and "round" the static labeling $\mathcal{L}(t_1)$ to a canonical one by going through the points one-by-one from left to right, and for each point with a non-canonical label doing the following. First move its label continuously to the left, stopping when the label hits a vertical line defined by

another label which is already canonical, or when the label is completely to the left of its point. This makes the label canonical along the x-axis. Now move the label upward in an analogous fashion until it is canonical along the y-axis. Having made $\mathcal{L}(t_1)$ canonical in this fashion, let $t_2 > t_1$ denote the first event time for this labeling. We then round $\mathcal{L}(t_2)$ to a canonical labeling in the same fashion, let $t_3 > t_2$ denote *its* first event, and continue on until reaching $t_m \geqslant b$ for some m. We now have a sequence of canonical labelings, and only need to show that we can intersperse them with label moves to transition from one to another without label overlap. To this end, consider the continuous label movement that occurred in the original dynamic labeling $\mathcal{L}$ during the interval (t_i, t_{i+1}), for some $i \in \{1, \ldots, m-1\}$. Now prepend the label motions used in the rounding of $\mathcal{L}(t_i)$ and append the label motions used in the rounding of $\mathcal{L}(t_{i+1})$. This results in overlap-free label motions between the two canonical labelings $\mathcal{L}(t_i)$ and $\mathcal{L}(t_{i+1})$. From that we can compute a sequence of discrete label moves by determining the order in which labels cross horizontal or vertical lines in this motion.

To conclude the proof, we have to show how all of the above can be modified for the remaining cases allowed by the problem definition.

Firstly, we shall deal with the other two label models. The case of the 4-position model is actually easier than that of the 4-slider model. In the 4-position model any static labeling is canonical. Transitions between them involve moving each label one position clockwise, counter-clockwise, or not at all, and may be done instantly. A canonical dynamic labeling is therefore automatically also a proper dynamic labeling. The case of the 2-slider model is simply a combination of the other two: it behaves like the 4-slider model along one axis, and like the 4-position model along the other.

Secondly, we have assumed so far that neither $\mathcal{L}(a)$ nor $\mathcal{L}(b)$ was given, and had to be guessed non-deterministically. If either or both are given, however, the procedure is much the same. If $\mathcal{L}(a)$ is given, then we simply use *it* instead of our own guess. When guessing the label moves before the first event, the label positions as given by $\mathcal{L}(a)$ will then also count as canonical. Similarly, if $\mathcal{L}(b)$ is given, the final label moves are guessed in such a way that the label positions of $\mathcal{L}(b)$ count as canonical and are the final result.

Lastly, we have assumed that no points arrive or depart in the interval (a, b). Suppose points *do* arrive and/or depart, at times $a = e_1, e_2, \ldots, e_\ell = b$. Simply apply the above separately on each of the consecutive time intervals $[e_1, e_2], [e_2, e_3], \ldots, [e_{\ell-1}, e_\ell]$. The static labeling $\mathcal{L}(e_i)$ for each $i \in \{2, \ldots, \ell - 1\}$ is then computed as output for the interval $[e_{i-1}, e_i]$ and used as input for the interval $[e_i, e_{i+1}]$. This merely requires dropping the points removed at e_i from the labeling, and guessing canonical label positions for the points added at e_i. □

4. Conclusion

We have examined the following dynamic point labeling problem. Given a set of points moving along continuous trajectories, and where points may be added and removed over time, is there a smooth function from time to static point labelings? For the 4-position, 2-slider, and 4-slider models we show this problem to be strongly PSPACE-complete. In addition, finding the maximum label size at which such a labeling does exist admits no PTAS, unless P = PSPACE. For the 4-position model a 2-approximation is the best that can be hoped for, and for the other models a 4/3-approximation. The PSPACE-completeness results also apply for the special case where the points are panned, rotated, or zoomed inside a fixed viewport. For this case our constructions have less "wiggle room", meaning our hardness-of-approximation results do not apply. The wiggle room is still non-zero, however, meaning that a PTAS would still imply P = PSPACE.

We have achieved these results using the non-deterministic constraint logic framework by Hearn and Demaine,[8] which is usually applied to the complexity of motion planning in general and solving (puzzle) games in particular. By placing point labeling in this framework we obtain a large variety of hardness results with just a single set of gadgets. Apart from the strong PSPACE-completeness of dynamic point labeling, we used the framework to show strong NP-completeness of static point labeling. This unifies known results for the 4-position and 4-slider models, as well as a new result for the 2-slider model, all into a single, conceptually simple proof.

It remains to examine other label models such as the 1- and 2-position models and the 1-slider model. For static labeling their corresponding decision problems are easily solved in polynomial time. On the other hand, the number-maximization problem is still NP-hard for them. Perhaps the complexity landscape of dynamic labeling is similar. Most important, perhaps, is the continued pursuit of approximation algorithms for optimization problems in dynamic point labeling. While there are heuristics for both number maximization[11] and free-label maximization,[12] no guaranteed approximation ratios have yet been achieved.

References

1. A. Wolff and T. Strijk. The Map Labeling Bibliography. `http://liinwww.ira.uka.de/bibliography/Theory/map.labeling.html`, 2009.
2. M. Formann and F. Wagner. A packing problem with applications to lettering of maps. In *Proc. 7th ACM Sympos. Comput. Geom. (SoCG'91)*, pages 281–288. 1991.
3. C. Iturriaga. *Map Labeling Problems*. Ph.D. thesis, University of Waterloo, Canada, 1999.
4. M. van Kreveld, T. Strijk, and A. Wolff. Point labeling with sliding labels. *Comput. Geom. Theory Appl.*, 13:21–47, 1999.
5. 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.
6. A. Gemsa, M. Nöllenburg, and I. Rutter. Consistent labeling of rotating maps. In *Proc.*

12th Internat. Sympos. Algorithms and Data Structures (WADS'11), pages 451–462. 2011.

7. R. A. Hearn and E. D. Demaine. PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation. *Theoret. Comput. Sci.*, 343(1–2):72–96, 2005.
8. R. A. Hearn and E. D. Demaine. *Games, Puzzles, and Computation.* A K Peters, 2009.
9. T. Biedl and G. Kant. A better heuristic for orthogonal graph drawings. *Comput. Geom. Theory Appl.*, 9(3):159–180, 1998.
10. W. J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *J. Comput. Syst. Sci.*, 4(2):177–192, 1970.
11. M. Vaaraniemi, M. Treib, and R. Westermann. Temporally coherent real-time labeling of dynamic scenes. In *Proc. 3rd Internat. Conf. Computing for Geospatial Research and Applications (COM.Geo'12), article no. 17.* 2012.
12. M. de Berg and D. H. P. Gerrits. Labeling moving points with a trade-off between label speed and label overlap. In *Proc. 21st European Sympos. Algorithms (ESA'13).* 2013. To appear.