


# Transitions in Dynamic Point Labeling

Thomas Depian ✉

Algorithms and Complexity Group, TU Wien, Austria

Guangping Li ✉ 

Algorithm Engineering Group, TU Dortmund, Germany

Martin Nöllenburg ✉ 

Algorithms and Complexity Group, TU Wien, Austria

Jules Wulms ✉ 

Algorithms and Complexity Group, TU Wien, Austria

---

## Abstract

The labeling of point features on a map is a well-studied topic. In a static setting, the goal is to find a non-overlapping label placement for (a subset of) point features. In a dynamic setting, the set of point features and their corresponding labels change, and the labeling has to adapt to such changes. To aid the user in tracking these changes, we can use morphs, here called *transitions*, to indicate how a labeling changes. Such transitions have not gained much attention yet, and we investigate different types of transitions for labelings of points, most notably *consecutive* transitions and *simultaneous* transitions. We give (tight) bounds on the number of overlaps that can occur during these transitions. When each label has a (non-negative) weight associated to it, and each overlap imposes a penalty proportional to the weight of the overlapping labels, we show that it is NP-complete to decide whether the penalty during a simultaneous transition has weight at most  $k$ . Finally, in a case study, we consider geotagged Twitter data on a map, by labeling points with rectangular labels showing tweets. We developed a prototype implementation to evaluate different transition styles in practice, measuring both number of overlaps and transition duration.

**2012 ACM Subject Classification** Theory of computation → Computational geometry; Human-centered computing → Geographic visualization

**Keywords and phrases** Dynamic labels, Label overlaps, Morphs, NP-completeness, Case study

**Digital Object Identifier** 10.4230/LIPIcs.GIScience.2023.2

**Related Version** Full Version: <https://arxiv.org/abs/2202.11562>

**Funding** *Guangping Li*: Funded by the Austrian Science Fund (FWF) under grant P31119.

*Jules Wulms*: Funded partially by the Austrian Science Fund (FWF) under grant P31119 and partially by the Vienna Science and Technology Fund (WWTF) under grant ICT19-035.

## 1 Introduction

Maps are ubiquitous in the modern world: from geographic to political maps, and from detailed road networks to schematized metro maps, maps are used on a daily basis. Advances in technology allow us to use digital maps on-the-fly and in a highly interactive fashion, by means of panning, zooming, and searching for map features. Besides changes induced by the user, maps can also change passively, for example automated panning during gps routing, or changing points of interest when visualizing time-varying geospatial (point) data.

Important features on a map are often labeled. Examples of such features are areas (such as countries and mountain ranges), curves (for example roads and rivers), and most importantly points (of interest). The aforementioned interactions force map features and their corresponding labels to change, by appearing, disappearing, or changing position. Instead of swapping between the map before and after such changes, we can use morphs, here called *transitions*, to allow the user to more easily follow changes in map features and labelings.



© Thomas Depian, Guangping Li, Martin Nöllenburg, and Jules Wulms;  
licensed under Creative Commons License CC-BY 4.0

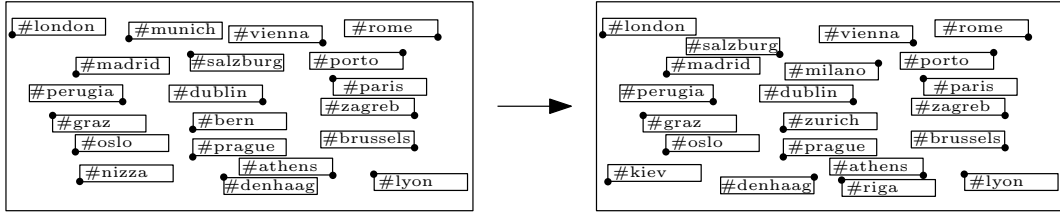
12th International Conference on Geographic Information Science (GIScience 2023).

Editors: Roger Beecham, Jed A. Long, Dianna Smith, Qunshan Zhao, and Sarah Wise; Article No. 2; pp. 2:1–2:19

Leibniz International Proceedings in Informatics



Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



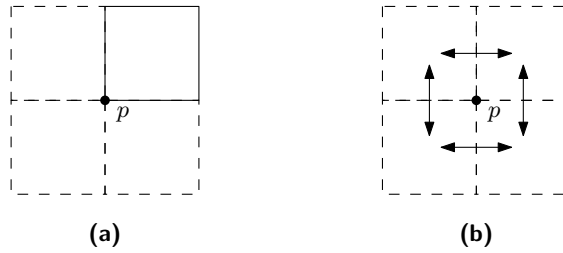
■ **Figure 1** A full visual scan of the individual labels is necessary to identify all changes [20].

Figure 1 shows why such transitions are important: even for two very similar point labelings, a lot of mental effort can be required to identify the differences.

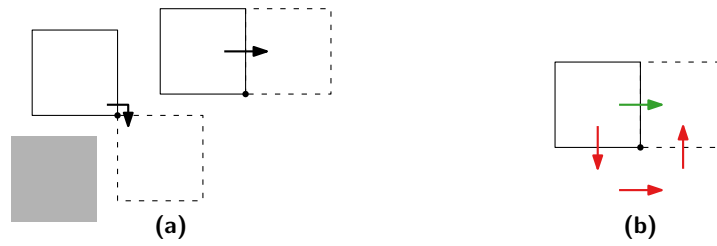
Automated map labeling is a well-researched topic within the geographic information science (GIS) and computational geometry community. In recent years, the GIS community has investigated the labeling of road networks [18], island groups [21], time-varying maps [2, 14], combining labeling with word clouds [5], and using human-in-the-loop approaches for labeling [13]. Algorithms have mainly focused on (the complexity of) computing labelings, in various static [1, 11, 22], interactive [3, 4, 12, 15], and dynamic or kinetic [6, 7, 9] settings.

In this paper we study transitions on maps that show point features  $P$  and their labels  $L$ . Let  $P$  be a finite point set in  $\mathbb{R}^2$ , where each point  $p_i \in P$  has a label  $l_i \in L$  associated to it. Labels are axis-aligned rectangles in the frequently used four-position model, that is, each point  $p_i$  has four possible candidate positions to place label  $l_i$  [11] (see Figure 2a). While labels are often modeled as arbitrary (axis-aligned) rectangles, we use squares with side length  $\sigma = 1$  for simplicity. In Appendix A we show how our results extend to arbitrary rectangles. A labeling  $\mathcal{L} \subseteq L$  of  $P$  consists of a set of pairwise non-overlapping labels, and can be drawn on a map conflict-free, by drawing only the labels in  $\mathcal{L}$  with their associated points. If the label  $l \in L$  for a point  $p \in P$  is not contained in  $\mathcal{L}$ , we do not draw  $p$  either.

Furthermore, we work in a dynamic setting, where points appear and disappear at different moments in time, and the set  $P$  changes only through additions and deletions: the data we consider later consists of geotagged tweets, for which we know only the location at the moment they are tweeted, and hence data points do not move. Every time changes are made to  $P$ , a new overlap-free labeling must be computed, thus resulting in a change from labeling  $\mathcal{L}_1$ , before the changes, to labeling  $\mathcal{L}_2$ , afterwards. In this paper we study different types of transitions from  $\mathcal{L}_1$  to  $\mathcal{L}_2$ . During such a transition, the individual labels are allowed to move in the sliding-position model [22] (see Figure 2b). Our aim is to find transitions that achieve optimization criteria, such as minimizing the number of overlaps during a transition, or minimizing the time required to perform a transition. To our knowledge, this is the first time transitions have been studied in this way.



■ **Figure 2** (a) The four candidate positions for label  $l$  of point  $p$ , with  $l$  placed in the top-right position. (b) Labels continuously move between candidate positions using the sliding-position model.



■ **Figure 3** (a) Minimizing overlaps by moving around the gray stationary label. (b) Minimizing duration by using a single movement along the green arrow, instead of moving along the red arrows.

**Problem description.** Given two (overlap-free) labelings  $\mathcal{L}_1$  and  $\mathcal{L}_2$ , we denote a transition between them with  $\mathcal{L}_1 \rightarrow \mathcal{L}_2$ . Such a transition consists of changes of the following types.

**Additions** If only label  $l_i$  of a feature point  $p_i$  must be added, we denote this by  $\mathcal{L}_1 \xrightarrow{A_i} \mathcal{L}_2$ .

**Removals** If only label  $l_i$  of a feature point  $p_i$  must be removed, we denote this by  $\mathcal{L}_1 \xrightarrow{R_i} \mathcal{L}_2$ .

**Movements** If only label  $l_i$  of a feature point  $p_i$  must change from its position in  $\mathcal{L}_1$  to a new position in  $\mathcal{L}_2$ , we denote this by  $\mathcal{L}_1 \xrightarrow{M_i} \mathcal{L}_2$ . Movements are unit speed and axis-aligned, in the sliding-position model. Note that a diagonal movement, as in Figure 3a (left), is composed of a horizontal and a vertical movement, and hence takes two units of time.

A label is *stationary* if it remains unchanged during a transition. Applying multiple transitions consecutively is indicated by chaining the corresponding transition symbols:  $\mathcal{L}_1 \xrightarrow{M_i M_j} \mathcal{L}_2$  denotes that label  $l_i$  moves before label  $l_j$ . Furthermore,  $\mathcal{L}_1 \xrightarrow{M} \mathcal{L}_2$  is a shorthand for applying all movement-transitions simultaneously. All these notions extend to additions and removals, using  $A$  and  $R$ , respectively, instead of  $M$ . A transition has no effect if no point must be transformed with the respective transition, e.g., even if there are no additions, the transition  $\mathcal{L}_1 \xrightarrow{A} \mathcal{L}_2$  is still applicable; it simply does not modify the labeling.

We aim to identify types of transitions that try to achieve the following goals.

**$\mathcal{G}_1$ – Minimize overlaps** While the two labelings are overlap-free, overlaps can occur during the transition from  $\mathcal{L}_1$  to  $\mathcal{L}_2$ . When too many overlaps happen at the same time, certain labels may (almost) completely disappear behind others during a transition, which defeats the purpose of the transition: allowing users to follow changes in the labeling. Thus, those overlaps should be avoided as much as possible, by, for instance, adjusting the movement direction of labels, as shown in Figure 3a.

**$\mathcal{G}_2$ – Minimize transition duration** Our main goal is to show a map in a (mostly) static state. However, we do not want to instantly swap  $\mathcal{L}_1$  for  $\mathcal{L}_2$ , since the user will have difficulties tracking all changes [20]. Though, a transition that takes too long can also cause users to lose attention [19]. Hence, we want transitions that can be completed in a short amount of time. This can be achieved by disallowing detours, as in Figure 3b, or by performing the changes simultaneously.

Note that ideally, one would also try to minimize the number of moving labels, as studies have showed that the amount of information humans can process is limited [17]. However, in this paper, we assume that we are given the new labeling  $\mathcal{L}_2$ , which thus dictates the labels that have to move. Therefore, we see the task of computing a *stable* labeling  $\mathcal{L}_2$ , i.e., one where only a few labels move, as an interesting research question in its own right.

Optimizing both goals  $\mathcal{G}_1$  and  $\mathcal{G}_2$  simultaneously is often impossible as there can be a trade-off: performing the transition as fast as possible to achieve  $\mathcal{G}_2$  often leads to unnecessary overlaps, while preventing as many overlaps as possible to achieve  $\mathcal{G}_1$  may require more time.

However, to work towards both  $\mathcal{G}_1$  and  $\mathcal{G}_2$ , we can perform all additions simultaneously, as well as all removals. Furthermore, if we perform removals before movements, and movements before the additions, we create free space for the movements, to reduce the number of overlaps without wasting time. Let  $X$  be an arbitrary way of performing all movements required to change from  $\mathcal{L}_1$  to  $\mathcal{L}_2$  (consecutively or simultaneously), then we can observe the following.

► **Observation 1.1.** *A transition of the form  $\mathcal{L}_1 \xrightarrow{RXA} \mathcal{L}_2$  aids in achieving both  $\mathcal{G}_1$  and  $\mathcal{G}_2$ .*

We introduce two overarching *transition styles* in this paper: *consecutive transitions* and *simultaneous transitions*. Each such transition style is a variant of the style  $RXA$ , as prescribed by Observation 1.1, and fills in the movement described by  $X$  in a unique way. For a consecutive transition the movement  $X$  consists of a sequence of label movements, whereas for a simultaneous transition we have  $X = M$ . These transition styles each incur different transition durations. Since we expect a trade-off between  $\mathcal{G}_1$  and  $\mathcal{G}_2$ , we specifically analyze the number of overlaps during transitions of the two styles.

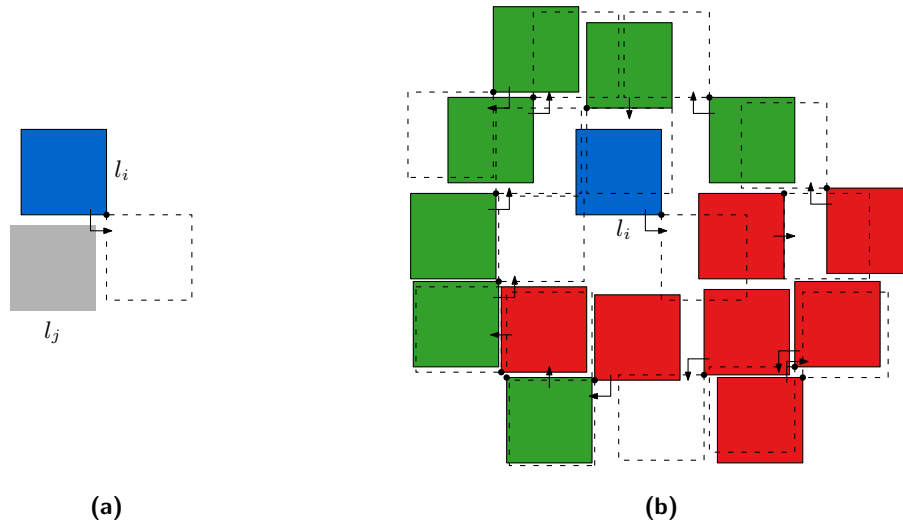
**Related work.** Our problem description resembles earlier work on point labeling, but it also has subtle differences. For example, the optimization criteria we care for, minimizing overlaps and time required for labels to move, were already investigated by de Berg and Gerrits [9]. They showed that there often is a clear trade-off between these criteria when dealing with moving labels. However, in their model, points are allowed to move (even during label movement), while our points are static, and change only through additions and deletions. Furthermore, in the PSPACE-hardness framework by Buchin and Gerrits [7] points are often static and only labels move. Hence, their dynamic labeling instances are similar to transitions. Though, a distinct difference is that labels must be allowed to move back and forth in the dynamic labeling instances of the hardness reduction. Since we disallow detours in transitions (see goal  $\mathcal{G}_2$ ), this reduction is not easily transferred to our setting.

Finally, our analysis of the number of overlaps in transitions draws multiple parallels with the analysis of topological stability, introduced in the framework for algorithm stability [16]. This framework provides various (mathematical) definitions of stability for algorithms on time-varying data: Intuitively, small changes in the input of an algorithm should lead to small changes in the output. Topological stability prescribes that the output changes continuously. The (topological) stability ratio of an algorithm then measures how close to optimum the stable output is: when an optimal solution undergoes a discrete change, a topologically stable output has to continuously morph through suboptimal solutions. Similarly, we analyze transitions with continuous movement of various styles. We then analyze how close to overlap-free a labeling is during a transition by counting overlaps.

**Contributions.** In Sections 2 and 3 we analyze the worst-case number of overlaps of consecutive and simultaneous transitions, respectively. In Section 3 we additionally consider instances where we associate weights to the labels (and to their overlaps) and prove that it is NP-hard to minimize the weight of overlaps in simultaneous transitions. Finally, in Section 4 we investigate in a case study how the transition styles perform on the described goals.

## 2 Consecutive Transitions

**Naive transitions.** Before we can propose more elaborate transition styles, we first evaluate the potential overlaps for a single label performing its movement. Figure 4a shows how only a single stationary square label can interfere with the moving label.



■ **Figure 4** (a) Since all labels are squares with side length  $\sigma$ , the moving blue label  $l_i$  can overlap only a single gray stationary label  $l_j$ . (b) The blue label  $l_i$  overlaps 14 other labels during the movement transitions. The green labels move before  $l_i$ , red labels move after  $l_i$ .

► **Lemma 2.1.** In  $\mathcal{L}_1 \xrightarrow{RM_i A} \mathcal{L}_2$ , where only label  $l_i$  moves, at most one overlap can occur.

**Proof.** As we perform removals before the movement and additions afterwards, we can guarantee that the start and end positions of label  $l_i$  are free. Thus any overlap can occur only during diagonal movement of  $l_i$ , when  $l_i$  moves from one candidate position in  $\mathcal{L}_1$ , to a non-adjacent candidate position in  $\mathcal{L}_2$ . Assume without loss of generality that  $l_i$  traverses the lower-left label position, when moving from top-left to bottom-right. Only a single other (stationary) label  $l_j$  can be positioned such that both  $\mathcal{L}_1$  and  $\mathcal{L}_2$  are overlap-free and the label overlaps with the area traversed by  $l_i$  (see Figure 4a). Any additional label overlapping the traversed area, without overlapping  $l_j$ , would overlap the start or end position of  $l_i$ . ◀

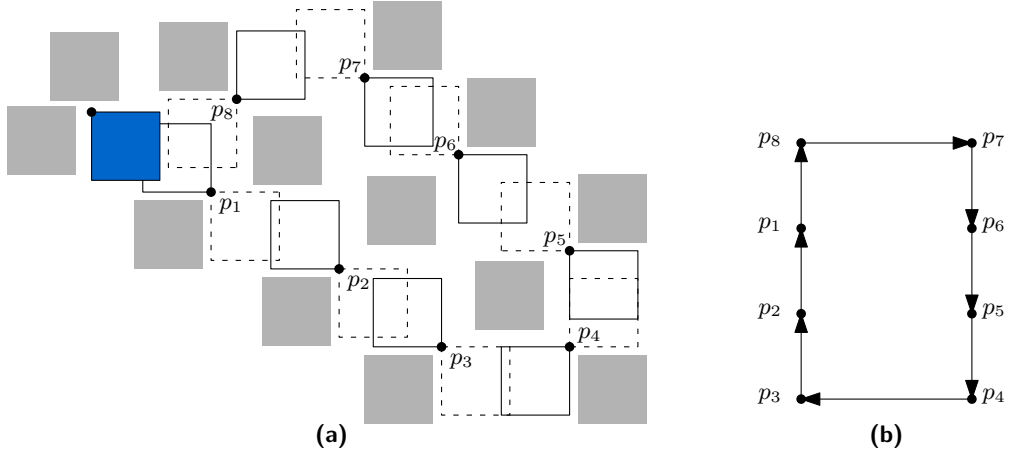
Next we consider an arbitrary order of all  $n$  moving labels in a transition. We define a conflict graph, which has a vertex for each moving label, and an edge between overlapping labels. With a packing argument we locally bound the degree of each of the  $n$  moving labels to 14 by considering the start, intermediate, and end position of such a label (these overlaps are achieved in Figure 4b). By the handshaking lemma this results in at most  $7n$  overlaps. For more details on the proof of Lemma 2.2 see the full version [10].

► **Lemma 2.2.** In  $\mathcal{L}_1 \xrightarrow{RM_1 \dots M_n A} \mathcal{L}_2$  at most  $7n$  overlaps can occur.

**DAG-based transitions.** To refine the naive approach, we model dependencies between movements in a *movement graph*, and use it to order movements and avoid certain overlaps.

► **Definition 2.3** (Movement graph). Let  $\mathcal{M} = \{M_1, \dots, M_n\}$  be a set of movements. Create for each movement  $M_i \in \mathcal{M}$  a vertex  $v_i$ , and create a directed edge from  $v_i$  to  $v_j$ ,  $v_i \rightarrow v_j$ , if some intermediate or end position of  $M_j$  overlaps with the start position of  $M_i$ , or the end position of  $M_j$  overlaps with some intermediate position of  $M_i$ : In both cases movement  $M_i$  should take place before movement  $M_j$ . If intermediate positions of  $M_i$  and  $M_j$  overlap, create the edge  $v_i \rightarrow v_j$ ,  $i < j$ . This results in the movement graph  $G_{\mathcal{M}}$  (see Figure 5).

A *feedback arc set* in a movement graph is a subset of edges that, when removed, breaks all cycles, resulting in a directed acyclic graph (DAG). We order movements using this DAG.



■ **Figure 5** (a) The blue label is added in this transition and forces  $n + m$  inevitable overlaps during movement ( $n = 8$  and  $m = 1$ ). Gray labels are stationary. (b) The corresponding movement graph.

► **Theorem 2.4.** *Movements in  $\mathcal{L}_1 \xrightarrow{RM_1 \dots M_n A} \mathcal{L}_2$  can be rearranged such that at most  $n + m$  overlaps occur, if  $G_{\mathcal{M}}$ , with  $\mathcal{M} = \{M_1, \dots, M_n\}$ , has a feedback arc set of size  $m$ .*

**Proof.** By Lemma 2.1, we know that at most one overlap occurs when moving a single label to a free end position. This leads to at most  $n$  overlaps for  $n$  consecutively moving labels, if no label moves to (or through) a position occupied by a label, which starts moving later.

Let  $G_{\mathcal{M}}$  be a movement graph with  $\mathcal{M} = \{M_1, \dots, M_n\}$ . There are two cases:

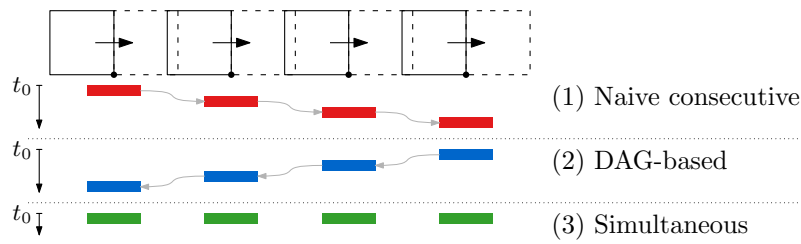
**Case (1)** If  $G_{\mathcal{M}}$  is acyclic, then handling all movements according to any topological ordering of the vertices of  $G_{\mathcal{M}}$  produces no additional overlaps.

**Case (2)** If  $G_{\mathcal{M}}$  contains cycles, then overlaps may be inevitable because each label in such a cycle wants to move to or through a position that is occupied by another moving label. Moreover, as the movements happen consecutively, one label in this cycle must move first and therefore may cause an overlap. Let  $m$  be the smallest number of edges that must be removed to break each cycle in  $G_{\mathcal{M}}$ , i.e., the size of a minimum feedback arc set  $S$ . As  $G_{\mathcal{M}}$  is cycle-free after removing  $S$ , case (1) applies and  $m$  additional overlaps suffice. ◀

We can see in Figure 5 that this bound is tight. Furthermore, it is not always necessary to perform all movements consecutively. We can observe that movements which are unrelated in  $G_{\mathcal{M}}$  can be performed simultaneously: when no overlap is possible, there is no edge in  $G_{\mathcal{M}}$ .

### 3 Simultaneous Transitions

Figure 6 shows three timelines of different transition styles, (1) a naive consecutive transition, (2) a DAG-based transition, and (3) a simultaneous transition. All transition styles start at some time  $t_0$  and the order of the movements of the labels for (1) and (2) is indicated with gray arrows. While (1) produces four overlaps and takes four units of time, (2) and (3) produce no overlaps, and (3) only takes a single unit of time. This shows that it is sometimes unnecessary to perform the movements consecutively to minimize overlaps. In this section, we investigate both how simultaneous movements influence the number of overlaps, and the complexity of minimizing overlaps.



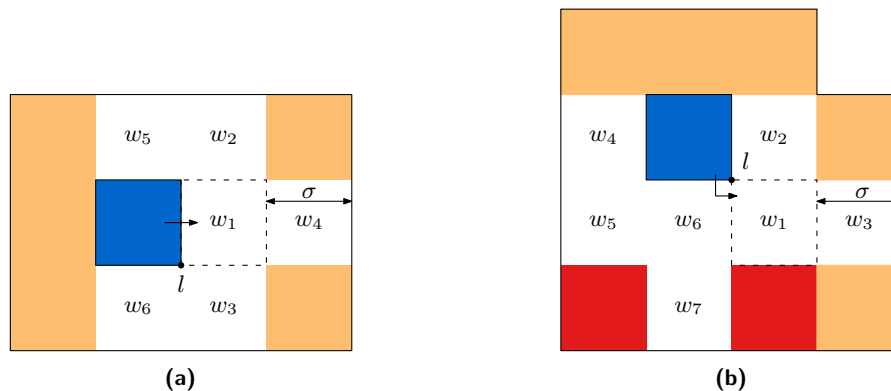
■ **Figure 6** Comparison of possible movement orderings with respect to  $\mathcal{G}_1$  and  $\mathcal{G}_2$ .

► **Theorem 3.1.** In  $\mathcal{L}_1 \xrightarrow{RMA} \mathcal{L}_2$  at most  $6n$  overlaps can occur, where  $n$  is the number of labels that must be moved, and all movements are performed at unit speed.

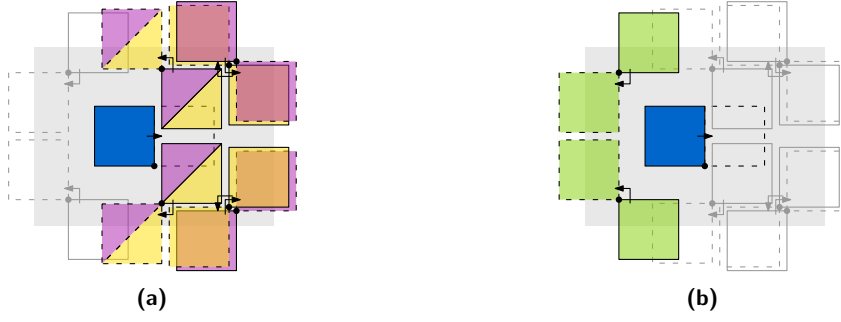
**Proof.** Let  $\sigma = 1$  denote the side length of a label. To show that the total number of overlaps is at most  $6n$ , we model the overlaps in a graph and consider the neighborhood of individual vertices. Let  $G$  be a conflict-graph where each vertex  $v_i$  corresponds to a label  $l_i$ . If two labels  $l_i$  and  $l_j$  overlap during the transition, we create an edge  $(v_i, v_j)$ , i.e., each edge corresponds to an overlap. Observe that each edge is adjacent to at least one moving label since two stationary labels cannot overlap. We proceed by evaluating in  $G$  the maximum possible degree of a moving label  $l$  and restrict ourselves to a  $\sigma$ -wide border around the bounding box of the movement area of  $l$ , that represents the area other labels must touch (before the transition) to overlap with  $l$ . We call this area the *overlapping region* of  $l$  and it is illustrated in Figure 7. Labels not intersecting the overlapping region of  $l$  by construction cannot overlap with  $l$ . We proceed by considering the two possible types of movements for  $l$ .

**Non-diagonal movement of  $l$ .** For a label  $l$  that performs a non-diagonal movement, the overlapping region is illustrated in Figure 7a. The light-orange area in the overlapping region indicates that the start position of a label overlapping  $l$  cannot lie solely in this area. If a label starts in the area behind  $l$ , then such a label would never overlap with  $l$ , since labels move simultaneously. The start position of labels overlapping  $l$  can neither overlap only the  $\sigma \times \sigma$  tiles diagonally adjacent to the end position of  $l$ , as the end position of those labels would overlap the end position of  $l$ , for any movement that allows the labels to overlap  $l$ .

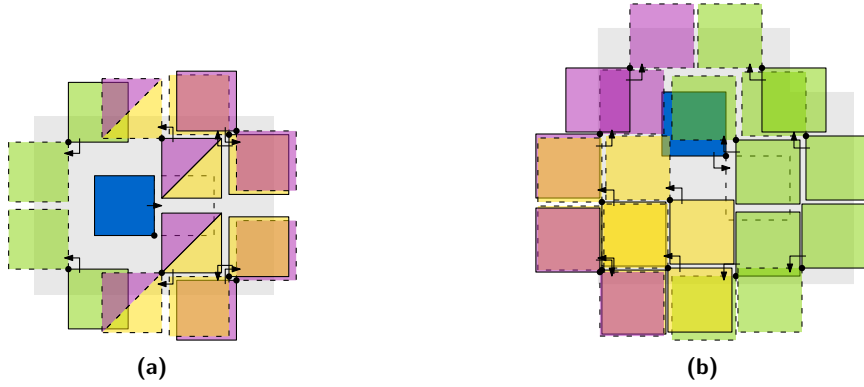
Next consider the remaining (white) area in the overlapping region, and see Figure 8 for our construction. Consider first the end position  $w_1$  of  $l$  and the three  $\sigma \times \sigma$  tiles  $w_2, w_3$  and  $w_4$  adjacent to it. The total height of  $w_1, w_2$  and  $w_3$  combined is  $3\sigma$ , and hence the



■ **Figure 7** Overlapping regions for (a) non-diagonal and (b) diagonal movement of the blue label  $l$ .



■ **Figure 8** Labels overlapping with the blue label  $l$  that are located on the white tiles (a)  $w_1 - w_4$  and (b)  $w_5 - w_6$ . The overlapping region of  $l$  is indicated in gray.



■ **Figure 9** Labels overlapping the blue label  $l$  that performs a (a) non-diagonal and a (b) diagonal movement. The overlapping region of  $l$  is indicated in gray.

start positions of at most four labels can be stacked vertically to overlap this area (see the labels with the color ■ in Figure 8a). Similarly,  $w_1$  and  $w_4$  have a combined width of  $2\sigma$  and height  $\sigma$ . Since the end position of  $l$  is adjacent to the start position of  $l$  we can put at most two labels horizontally next to each other in this area, while keeping  $\mathcal{L}_1$  overlap free. However, as the height is  $\sigma$  we can stack at most two layers of such labels vertically (the ■ labels in Figure 8a). As a result,  $w_1, w_2, w_3$  and  $w_4$  can together overlap with at most six start positions of other labels. Each label results in at most one overlap, and there is a movement direction for each label that achieves such an overlap, as shown in Figure 8a.

Now consider the  $\sigma \times \sigma$  tiles  $w_5$  and  $w_6$  above and below the start position of  $l$ , respectively. We can place two labels, the ones colored ■ in Figure 8b, such that their start positions overlap either of  $w_5$  and  $w_6$ . For example, for  $w_5$  such labels can move diagonally down-left, to overlap  $l$ . In this case, it is impossible for a label overlapping  $w_6$  to both overlap  $l$  and have an overlap-free end position. Conversely, we can place one label on  $w_5$  and  $w_6$  each and allow them both to move towards  $l$ , while ensuring overlap-free end positions (see Figure 8b). Observe that it is impossible to place two labels on both  $w_5$  and  $w_6$  in the latter case, as the vertical positioning that ensures overlap-free end positions of the labels, requires the labels to start farther from  $l$ . Those labels have to move a vertical distance of at least  $\sigma/2$  to reach  $l$ , and hence also require a horizontal overlap of at least  $\sigma/2$  with the start position of  $l$ , as  $l$  will have moved  $\sigma/2$  rightwards before the other labels reach the start position of  $l$ . Thus, at most eight labels can overlap with  $l$ , and consequently the degree of the corresponding vertex is bounded by eight. See Figure 9a for the complete situation.



**Diagonal movement of  $l$ .** For diagonal movements, we consider w.l.o.g. the case where  $l$  performs a diagonal movement from top-left to bottom-right through the bottom-left corner. The overlapping region enlarges, as shown in Figure 7b. We can again eliminate the light-orange areas, as they mark areas that the start position of other labels cannot overlap exclusively, if they should overlap with  $l$ . As before, these areas are located behind the start position of  $l$ , and diagonally adjacent to the end position of  $l$ . The red areas are also eliminated, see the full version [10] for more details. We now repeat the process of filling the remaining (white)  $\sigma \times \sigma$  tiles,  $w_1$  to  $w_7$ , with start positions for labels that can overlap with  $l$  during movement. The analysis is very similar to the non-diagonal case, with one exception:  $l$  can overlap with one stationary label, which can now occupy  $w_6$ . We again do a case distinction on the possible label placements, showing that at most nine moving labels and one stationary label can overlap with  $l$ , or at most 12 moving labels can overlap with  $l$ . The upper bound of 12 overlaps for one label is tight (as shown in Figure 9b). See the full version [10] for the remaining details of this case.

**Deriving an upper bound.** To find an upper bound on the number of overlaps, consider the subgraph  $G[V_M]$  induced by the set  $V_M$  of vertices that represent moving labels. The degree of one such vertex in  $G[V_M]$ , which represents a label  $l$ , is bounded by nine, in case  $l$  moves diagonally and a stationary label is present on the intermediate position of  $l$ , or by 12, otherwise. Both these bounds are higher than in the non-diagonal movement case, which would result in a degree of at most eight. Hence, in the worst case we have a degree sum between  $9n$  and  $12n$ , respectively, since  $|V_M| = n$ . By the handshaking lemma, we then have between at most  $\lceil \frac{9}{2} \rceil n$  and  $6n$  edges in  $G[V_M]$ , respectively.

If we consider the original graph  $G$ , we can observe that it differs from  $G[V_M]$ , in terms of edges, only by the edges that are incident to a moving label and a stationary label. This means that the former case may result in more overlaps: As we have seen in one case of the above proof, and due to Lemma 2.1, a moving label can overlap with at most one stationary label. Since each of the edges in  $E(G) \setminus E(G[V_M])$  is incident to exactly one vertex that represents a moving label, and we have  $n$  of such labels,  $|E(G) \setminus E(G[V_M])|$  is bounded by  $n$  and consequently, we have at most  $(\lceil \frac{9}{2} \rceil + 1)n$  overlaps with the stationary label present. However, in the worst case we still have an upper bound of at most  $6n$  overlaps. ◀

### 3.1 Complexity of Computing Simultaneous Transitions

In this section, we show that it is NP-complete to minimize the number of overlaps in a weighted  $\mathcal{L}_1 \xrightarrow{RMA} \mathcal{L}_2$ -transition by choosing the direction of diagonal movements.

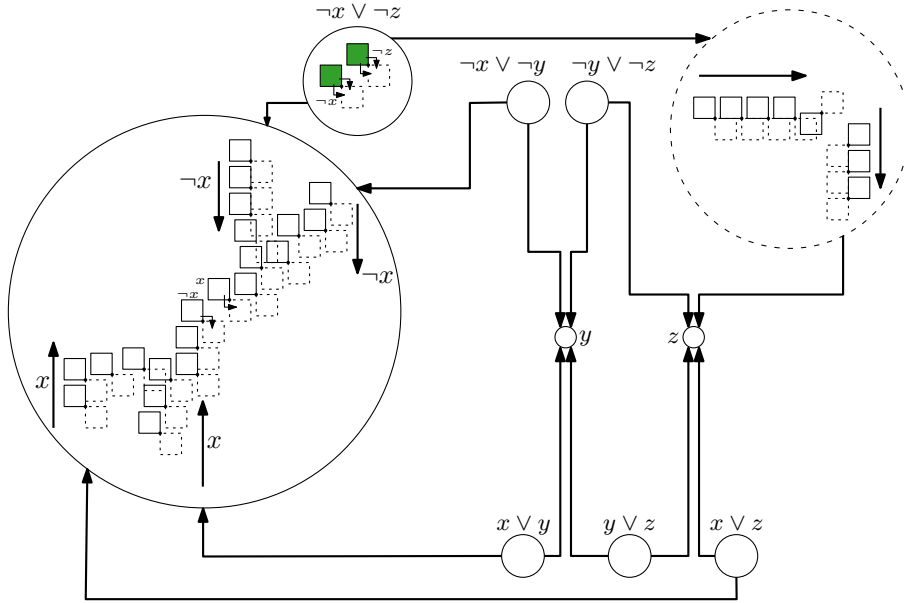
► **Definition 3.2** (Weighted Transition). *Let  $\mathcal{L}_1 \xrightarrow{\Sigma} \mathcal{L}_2$  be a transition, where  $\Sigma$  denotes an arbitrary transition style of additions, movements, and removals, and let  $w$  be a weight function that assigns to each label  $l \in L$  a non-negative weight  $w(l) \in \mathbb{R}_0^+$ . A weighted transition  $\mathcal{L}_1 \xrightarrow[\Sigma]{w} \mathcal{L}_2$  performs  $\mathcal{L}_1 \xrightarrow{\Sigma} \mathcal{L}_2$ , but when two labels  $l_i$  and  $l_j$  overlap, a penalty of weight  $w(l_i) \cdot w(l_j)$  is introduced. The total penalty  $W$  is equal to the sum of penalty weights.*

► **Problem 3.3.** *Given a weighted transition  $\mathcal{L}_1 \xrightarrow[\Sigma]{w} \mathcal{L}_2$  and  $k \in \mathbb{R}_0^+$ , can we assign a movement direction to each diagonal movement such that the total penalty  $W$  is at most  $k$ ?*

We sketch the proof of Theorem 3.4 here, details can be found in the full version [10].

► **Theorem 3.4.** *It is NP-complete to decide whether  $W$  is at most  $k$  for  $\mathcal{L}_1 \xrightarrow[\Sigma]{w} \mathcal{L}_2$ .*

**Proof sketch.** Given a movement direction for each label, it is easy to check whether  $W$  is at most  $k$  by considering each pair of labels and checking for overlaps. Hence Problem 3.3 is contained in NP. For NP-hardness, we reduce from an instance  $F$  of PLANAR MONOTONE MAX 2-SAT [8]. Figure 10 gives an overview of the required gadgets. Clause and variable gadgets consist of two opposing labels at their core, corresponding, respectively, to the assignments of the two literals in a clause, or the binary choice for a variable. For an unsatisfied clause, an overlap occurs inside the clause gadget, whenever both labels move towards each other (inwards). The corresponding labels have weight one, and hence such an overlap would incur a penalty of weight one. A variable gadget has two opposing labels for setting the variable to *true* or *false*. Choosing a movement direction outward from the variable gadget, for example on the “true”-side, will cause a domino effect, propagating towards the gadgets of clauses with negative occurrences of this variable. There it results in inward movement, and hence this corresponds to setting the variable to not be false (and thus be true). Choosing the outward movement for both variable states is never beneficial: that variable is neither true nor false. The movement directions chosen in the variable gadgets are propagated to the appropriate clauses using the (planar) embedding of the incidence graph of  $F$ . All labels outside of clause gadgets have weight  $n + 1$  and hence producing an overlap outside of a clause gadget will result in a large penalty of weight greater than  $n$ . As such, we either have movement directions that produce a total penalty of at most  $k$  for some positive  $k < n$ , and overlaps correspond to unsatisfied clauses, or we have a total penalty of at least  $n$ , and no clauses can be satisfied (or the variable assignment is inconsistent). Thus,  $n - k$  clauses are satisfiable in  $F$ , if and only if we have  $k$  overlaps in our reduced instance. ◀



■ **Figure 10** Reduced instance for the formula  $F = (\neg x \vee \neg z) \wedge (\neg x \vee \neg y) \wedge (\neg y \vee \neg z) \wedge (x \vee y) \wedge (y \vee z) \wedge (x \vee z)$ . The weight of white and green labels is  $n + 1$  and 1, respectively.

Figure 10 shows an example of the complete setup. There we reduce the formula  $F = (\neg x \vee \neg z) \wedge (\neg x \vee \neg y) \wedge (\neg y \vee \neg z) \wedge (x \vee y) \wedge (y \vee z) \wedge (x \vee z)$  onto our problem. Circles with solid borders indicate the individual parts of the formula, while the dashed circle shows part of a transportation gadget. The big empty circles represent the clauses and small empty

■ **Table 1** The Keywords and #Hashtags we used to query the tweets.

corona	#corona	covid	#covid	covid19	#covid19	covid-19
vaccine	#vaccine	quarantine	#quarantine	lockdown	#lockdown	moderna
outbreak	#outbreak	immune	#immune	immunity	#immunity	biontech
who	#who	desease	#desease	masks	#masks	pfizer
pandemic	#pandemic	mutation	#mutation	ffp2	#ffp2	
#StaySave	#StayAtHome	#FlattenTheCurve		astrazeneca	johnson & johnson	

circles represent the variables. The arrows outside circles represent the transportation-gadgets that connect the individual parts according to the direction of the arrows. As there exists no variable assignment which satisfies  $F$ , we cannot achieve  $W = k = 0$  but must encounter at least one overlap (and hence  $W \geq 1$ ). This overlap occurs, for example, in the clause gadget for  $(\neg x \vee \neg z)$  to achieve  $W = 1$ . Note that if we would try to resolve this overlap by, for instance, setting  $x$  to true and false at the same time, an overlap with penalty  $(n + 1)^2 = 49$  would occur, for example, in the variable-gadget for the variable  $x$ .

## 4 Case Study: Twitter Data

We implemented a prototype to analyze how the transition styles perform in a practical setting. In this prototype, we show geotagged tweets (Section 4.1) as rectangular labels on an interactive map (Section 4.3). In order to show an appropriate amount of information inside the labels, we use uniform-sized axis-aligned rectangles for the labels instead of squares. As all labels will have the same size, our theoretical results derived for square labels with side length 1 carry over to this model (by scaling horizontally). Finally, we measure the number of overlaps and the transition duration of the presented transition styles (Section 4.4).

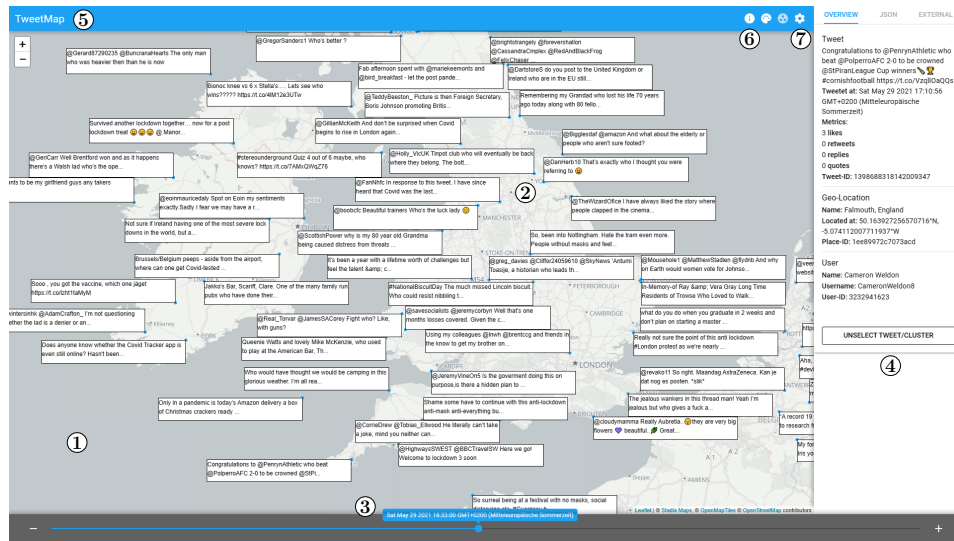
### 4.1 Dataset

For our dataset, we queried 100,000 geotagged tweets related to the COVID-19 pandemic during the month of May 2021, see Table 1. After filtering and cleaning this dataset, 99,982 usable tweets were left. A tweet will be represented as a point  $p \in P$ , and as its label we display parts of the *Tweet Text* in a rectangular box, possibly truncating it if it is too long. For the spatial and temporal property we use the *Tweet Location* and *Tweet Date and Time* fields, respectively. The former field is a location attached to a tweet, which will determine the coordinates of the point  $p \in P$ . This is not necessarily a concrete location, but can be a (rectangular) area on the map. We choose an arbitrary location inside this area to prevent artificial cluster creation. The latter field defines the date and time the tweet was posted.

### 4.2 Dynamic Labeling Model

The dataset described above has spatiotemporal properties: each point  $p \in P$  has a location and a time associated to it. Starting from this point in time, we consider  $p$  (and its associated tweet) *relevant* for three hours. The relevant tweets at a particular *time of interest* will form the set  $P$  of points that we want to label. Changes to the time of interest (dynamically) alter the set  $P$  of relevant tweets through additions and removals. Furthermore, in our implementation not all points in  $P$  will be in view at all times: For example, when the user zooms in on a particular part of the map, some points will be outside the view port. In such cases, we label only the subset  $S \subseteq P$  of points that are inside the view port.

## 2:12 Transitions in Dynamic Point Labeling



■ Figure 11 Screenshot of the prototype.

### 4.3 Implementation Details

The prototype computes a labeling in the four-position model of the relevant points  $P$ . Figure 11 shows a screenshot of our prototype. The main view area (①), in the center of the screen, shows a map and a labeling overlay. Furthermore, it contains blue dots (②) indicating the locations of the subset  $S \subseteq P$ . Below the map, the time slider (③) shows the currently selected time of interest. The side drawer on the right (④) shows further information of a tweet, if the user selects one. The top bar (⑤) allows the user to retrieve additional information about the map (⑥) and alter its state using the cogwheel (⑦).

The user can interact with the prototype by means of *panning* and *zooming* the map, as well as changing the time of interest by using the *time slider*. Panning is done by dragging the map using the mouse, while zooming is controlled using either the mouse wheel or the zoom indicators in the upper-left corner of the map. Zoom level changes step-wise. The time of interest is changed by dragging the indicator in the time slider, or using the + and – buttons on either side of the slider. Panning and zooming change the subset  $S \subseteq P$  of points in view, while changes to the time of interest alter the relevant points  $P$ .

**Computing a labeling.** For a given subset  $S \subseteq P$  of tweets that are both relevant and in view, we compute a labeling as follows. We create a conflict graph of labels for  $S$  and use a simple greedy approximation algorithm for a MAXIMAL INDEPENDENT SET  $I$  in this graph: iteratively add a minimum-degree vertex to  $I$ , that shares no edge with vertices in  $I$ .

When the user now interacts with the map, we again perform the same algorithm to find a new labeling, but we use two simple heuristics to improve the stability of the labeling. The first heuristic is based on desideratum D1 from [3]. Among other things, it proposes that the same labels should remain visible when zooming. To achieve this, we remove all neighbors of the previously shown labels in the conflict graph, to ensure they are picked again.

The second heuristic attempts to prevent unnecessary changes in the labeling: Let  $I_1$  be the subset of labeled points that remained relevant and visible after panning/zooming/time change, and let  $I_2$  be the newly computed set of points to be labeled. If  $I_2$  is less than 2% larger than  $I_1$ , then we simply keep the labeling of  $I_1$  instead of swapping to a labeling of  $I_2$ .

When the subset  $S \subseteq P$  of relevant tweets in view is changed, through panning or zooming, or when  $P$  is dynamically altered by changes in the time of interest, our prototype will trigger a transition. Let  $\mathcal{L}_1, \mathcal{L}_2$  be the computed labelings before and after the change, respectively. Our prototype supports naive, DAG-based, and simultaneous transitions for  $\mathcal{L}_1 \rightarrow \mathcal{L}_2$ . In each of these transition styles, a removal, an addition, or a movement of a single label has a duration of one second. A diagonal movement is split up into two non-diagonal movements with a duration of one second each, starting with the horizontal movement.

**Naive transitions.** Movements in this transition are performed consecutively in arbitrary order. Their order is based on the order in which we recognize the need for a movement.

**DAG-based transitions.** The movements in DAG-based transitions are also performed consecutively, though ordered according to a topological ordering of the movement graph  $G_{\mathcal{M}}$ . If  $G_{\mathcal{M}}$  contains cycles, we remove the vertex with the lowest in-degree and first move the label of the removed vertex. Additionally, we perform unrelated movements in  $G_{\mathcal{M}}$  simultaneously.

**Simultaneous transitions.** The movements in this transition are all performed simultaneously, immediately after the removals. The direction of diagonal movement is not optimized for minimum overlaps. Instead, we move horizontally first, to create a more uniform transition.

**Implementation.** The prototype is a three-tier-architecture, consisting of a *graph-database* (Neo4j) storing the tweets together with their potential label candidates, an *application tier* (Java Play Framework) computing the (new) labeling, and the *presentation tier* (Vue.js, Leaflet, and GreenSock) with which the user can interact and which visualizes the transitions.

In our case study we measured the running times of the individual components, which we report in Appendices B.1 and B.2. We can see that the majority of the time in the back-end (between 60% and 85%) is spent on querying the database, while the remaining parts run in less than 150ms in nearly all investigated cases. Computing the transitions in the front-end takes on average below 10ms, and never more than 20ms, which is negligible.

## 4.4 Measuring Transition Time and Number of Overlaps

In our case study, we use our prototype to simulate twelve interaction settings in six scenarios. The different scenarios we use in our case study are described in Table 2. In the first setting, we use different interactions depending on the scenario we consider. For each interaction type, we interact with the prototype by applying the following sequence of operations.

■ **Table 2** The different scenario states of the case study.

Scenario Name	Interaction	Map center		Zoom level	Time of interest
		Longitude	Latitude		
Italy	(a)	14.45	41.30	7	2021-05-29T13:20:00
Lausanne	(b)	6.37	46.45	7	2021-05-30T10:30:00
Leeds	(b)	-1.60	53.44	7	2021-05-29T13:00:00
Los Angeles	(c)	-117.78	33.84	9	2021-05-30T03:15:00
New Delhi	(a)	71.18	30.20	7	2021-05-29T08:30:00
Sao Paulo	(c)	-45.00	-20.65	7	2021-05-29T02:30:00

**Interaction (a):** (1) Increase the time of interest by 30 minutes, (2) zoom in by one zoom level with the help of the zooming indicators, (3) increase the latitude of the map’s center by 0.28 using the settings, and (4) increase the time of interest by five minutes with the + button next to the time slider.

**Interaction (b):** Interaction (a) in reverse order: (1) Increase the time of interest by five minutes with the + button next to the time slider, (2) increase the latitude of the map’s center by 0.28 using the settings, (3) zoom in by one zoom level with the help of the zooming indicators, and (4) increase the time of interest by 30 minutes

**Interaction (c):** (1) Zoom in by one zoom level with the help of the zooming indicators, *decrease* the time of interest by five minutes with the – button next to the time slider, (3) *decrease* the *longitude* of the map’s center by 1.7 using the settings, and (4) increase the time of interest by 20 minutes.

In Table 3, we report an overview of the most important results and refer to Appendix B for additional measurements. Simultaneous transitions are the fastest, and the naive transitions the slowest. Noteworthy is that the duration of the DAG-based transitions is close to the simultaneous transitions. This suggests that there is significant benefit in simultaneously performing movements that are unrelated in  $G_{\mathcal{M}}$ . Furthermore, we can see that the DAG-based transitions produce around half as many overlaps as the other styles, on average less than one overlap in each setting. Simultaneous transitions cause a similar number of overlaps as naive transitions, but on average the total number of overlaps is slightly better.

This case study shows that DAG-based transitions find a good compromise between the number of overlaps ( $\mathcal{G}_1$ ) and the duration of the transitions ( $\mathcal{G}_2$ ). However, simultaneous transitions are more appealing, if one favours faster transitions over the number of overlaps. Hence, we see a clear trade-off between the number of overlaps and transition duration, similar to previous work [9]. Videos of the different transition styles can be found online<sup>1</sup>.

<sup>1</sup> Link to videos: [https://osf.io/hnsvu/?view\\_only=7703ba40643440f8958a9b0120dc32f0](https://osf.io/hnsvu/?view_only=7703ba40643440f8958a9b0120dc32f0)

■ **Table 3** Evaluation results for each transition style in each setting, best scores per row are bold.

Scenario	Naive transitions				DAG-based transitions				Simultaneous transitions			
	#Overlaps			Duration [s]	#Overlaps			Duration [s]	#Overlaps			Duration [s]
	Avg.	Tot.	Max		Avg.	Tot.	Max		Avg.	Tot.	Max	
Italy, 1	0,40	2	6,50	2,20	<b>0,20</b>	<b>1</b>	4,49	1,40	0,60	3	<b>2,50</b>	<b>1,00</b>
Italy, 2	0,16	6	6,51	1,62	<b>0,05</b>	<b>2</b>	5,50	1,30	0,24	9	<b>2,50</b>	<b>1,08</b>
Lausanne, 1	<b>0,40</b>	<b>2</b>	16,49	6,50	<b>0,40</b>	<b>2</b>	4,51	2,30	<b>0,40</b>	<b>2</b>	<b>2,51</b>	<b>1,50</b>
Lausanne, 2	1,43	53	18,50	5,27	<b>0,78</b>	<b>29</b>	9,49	3,24	1,19	44	<b>2,50</b>	<b>1,78</b>
Leeds, 1	1,40	7	23,50	9,10	<b>0,80</b>	<b>4</b>	5,49	2,09	1,00	5	<b>2,49</b>	<b>1,10</b>
Leeds, 2	1,03	38	16,50	4,81	<b>0,59</b>	<b>22</b>	7,50	2,92	0,65	24	<b>2,51</b>	<b>1,78</b>
Los Angeles, 1	1,00	5	25,49	9,30	<b>0,40</b>	<b>2</b>	4,50	2,30	<b>0,40</b>	<b>2</b>	<b>2,49</b>	<b>1,49</b>
Los Angeles, 2	0,43	16	8,51	2,55	<b>0,30</b>	<b>11</b>	4,50	1,88	0,38	14	<b>2,50</b>	<b>1,53</b>
New Delhi, 1	1,20	6	22,48	10,80	<b>0,60</b>	<b>3</b>	8,49	4,60	1,40	7	<b>2,50</b>	<b>2,00</b>
New Delhi, 2	0,59	22	12,50	3,54	<b>0,27</b>	<b>10</b>	5,50	2,21	0,46	17	<b>2,51</b>	<b>1,65</b>
Sao Paolo, 1	0,20	1	25,50	9,70	<b>0,00</b>	<b>0</b>	8,49	3,30	0,60	3	<b>2,50</b>	<b>1,50</b>
Sao Paolo, 2	0,41	15	13,50	2,28	<b>0,24</b>	<b>9</b>	8,49	1,69	0,38	14	<b>2,50</b>	<b>1,20</b>
Avg. Setting 1	0.77	3.83	19.99	7.93	<b>0.40</b>	<b>2.00</b>	5.99	2.66	0.73	3.67	<b>2.50</b>	<b>1.43</b>
Avg. Setting 2	0.68	25.00	12.67	3.35	<b>0.37</b>	<b>13.83</b>	6.83	2.21	0.55	20.33	<b>2.51</b>	<b>1.50</b>



## 5 Conclusion

In this paper we performed a first investigation into the number of overlaps produced by transitions on labelings of points, and started by proving tight upper bounds for various transition styles. In addition, we implemented the transition styles in a prototype and performed a case study that revealed the need for sophisticated transition styles that find a good compromise between the number of overlaps and the duration of a transition. We see this paper as a first step towards understanding such transitions in point labeling. Therefore we have many open questions for future work, such as:

- Should we develop new transition styles or improve the existing ones? Can we utilize more structured movement, like performing all movements in the same direction simultaneously?
- Is it sensible to try to formalize more perception-oriented desiderata for transitions, such as the symmetry of transitions or the traceability of labels?
- Is choosing label directions in simultaneous transitions still NP-hard with unit weights?
- Can we compute a *stable* labeling  $\mathcal{L}_2$ , that minimizes the number of moving labels?

---

## References

- 1 Pankaj K. Agarwal, Marc J. van Kreveld, and Subhash Suri. Label placement by maximum independent set in rectangles. *Comput. Geom.*, 11(3-4):209–218, 1998. doi:10.1016/S0925-7721(98)00028-5.
- 2 Lukas Barth, Benjamin Niedermann, Martin Nöllenburg, and Darren Strash. Temporal map labeling: a new unified framework with experiments. In *Proc. 24th SIGSPATIAL*, pages 1–10, 2016. doi:10.1145/2996913.2996957.
- 3 Ken Been, Eli Daiches, and Chee-Keng Yap. Dynamic map labeling. *IEEE Trans. Vis. Comput. Graph.*, 12(5):773–780, 2006. doi:10.1109/TVCG.2006.136.
- 4 Ken Been, Martin Nöllenburg, Sheung-Hung Poon, and Alexander Wolff. Optimizing active ranges for consistent dynamic map labeling. *Comput. Geom.*, 43(3):312–328, 2010. doi:10.1016/j.comgeo.2009.03.006.
- 5 Sujoy Bhore, Robert Ganian, Guangping Li, Martin Nöllenburg, and Jules Wulms. Worbel: Aggregating point labels into word clouds. In *Proc. 29th SIGSPATIAL*, pages 256–267, 2021. doi:10.1145/3474717.3483959.
- 6 Sujoy Bhore, Guangping Li, and Martin Nöllenburg. An Algorithmic Study of Fully Dynamic Independent Sets for Map Labeling. In *Proc. 28th ESA*, pages 19:1–19:24, 2020. doi:10.4230/LIPIcs.ESA.2020.19.
- 7 Kevin Buchin and Dirk H. P. Gerrits. Dynamic Point Labeling is Strongly PSPACE-Complete. *Int. J. Comput. Geom. Appl.*, 24(4):373, 2014. doi:10.1142/S0218195914600127.
- 8 Kevin Buchin, Valentin Polishchuk, Leonid Sedov, and Roman Voronov. Geometric Secluded Paths and Planar Satisfiability. In *Proc. 36th SoCG*, pages 24:1–24:15, 2020.
- 9 Mark de Berg and Dirk H. P. Gerrits. Labeling Moving Points with a Trade-Off between Label Speed and Label Overlap. In *Proc. 21st ESA*, pages 373–384, 2013. doi:10.1007/978-3-642-40450-4\_32.
- 10 Thomas Depian, Guangping Li, Martin Nöllenburg, and Jules Wulms. Transitions in Dynamic Map Labeling, 2022. doi:10.48550/arXiv.2202.11562.
- 11 Michael Formann and Frank Wagner. A packing problem with applications to lettering of maps. In *Proc. 7th SoCG*, pages 281–288, 1991.
- 12 Andreas Gemsa, Martin Nöllenburg, and Ignaz Rutter. Consistent Labeling of Rotating Maps. *J. Comput. Geom.*, 7(1):308–331, 2016. doi:10.20382/jocg.v7i1a15.
- 13 Fabian Klute, Guangping Li, Raphael Löffler, Martin Nöllenburg, and Manuela Schmidt. Exploring Semi-Automatic Map Labeling. In *Proc. 27th SIGSPATIAL*, pages 13–22, 2019. doi:10.1145/3347146.3359359.

- 14 Filip Krumpel. Labeling points of interest in dynamic maps using disk labels. In *Proc. 10th GIScience*, pages 8:1–8:14, 2018. doi:10.4230/LIPIcs.GISCIENCE.2018.8.
- 15 Chung-Shou Liao, Chih-Wei Liang, and Sheung H. Poon. Approximation algorithms on consistent dynamic map labeling. *Theor. Comput. Sci.*, 640:84–93, 2016. doi:10.1016/j.tcs.2016.06.006.
- 16 Wouter Meulemans, Bettina Speckmann, Kevin Verbeek, and Jules Wulms. A Framework for Algorithm Stability and Its Application to Kinetic Euclidean MSTs. In *Proc. 13th LATIN*, pages 805–819, 2018. doi:10.1007/978-3-319-77404-6\_58.
- 17 George A. Miller. The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information. *Psychological review*, 63(2):81–97, 1956. doi:10.1037/h0043158.
- 18 Benjamin Niedermann and Martin Nöllenburg. An algorithmic framework for labeling road maps. In Jennifer A. Miller, David O’Sullivan, and Nancy Wiegand, editors, *Proc. 9th GIScience*, pages 308–322, 2016. doi:10.1007/978-3-319-45738-3\_20.
- 19 Jakob Nielsen. *Usability Engineering*. Academic Press, 1993.
- 20 Ronald A. Rensink, John K. O’Regan, and James J. Clark. To See or not to See: The Need for Attention to Perceive Changes in Scenes. *Psychological Science*, 8(5):368–373, 1997. doi:10.1111/j.1467-9280.1997.tb00427.x.
- 21 Arthur van Goethem, Marc J. van Kreveld, and Bettina Speckmann. Circles in the water: Towards island group labeling. In *Proc. 9th GIScience*, pages 293–307, 2016. doi:10.1007/978-3-319-45738-3\_19.
- 22 Marc J. van Kreveld, Tycho Strijk, and Alexander Wolff. Point labeling with sliding labels. *Comput. Geom.*, 13(1):21–47, 1999. doi:10.1016/S0925-7721(99)00005-X.

## A Arbitrary Rectangle Labels

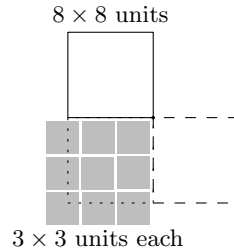
While in the main text we considered only square labels, point labelings often use arbitrary rectangles. If we allow our labels to be arbitrary rectangles, then it is no longer guaranteed that only one (stationary) label can overlap with the area traversed by the moving label. If we assume that the label with the largest side width (some  $\sigma_{x_{\max}}$ ) must perform a diagonal movement, we can align  $\frac{\sigma_{x_{\max}}}{\sigma_{x_{\min}}}$  stationary labels with a width of  $\sigma_{x_{\min}}$ , the smallest label width in our map, on the horizontal edge of the traversed area. As one label can always extend out of that traversed area without resulting in an invalid labeling  $\mathcal{L}_1$  or  $\mathcal{L}_2$ , we can put up to  $\lceil \frac{\sigma_{x_{\max}}}{\sigma_{x_{\min}}} \rceil$  labels next to each other in the  $x$ -direction. The same holds for the  $y$ -axis, with maximum and minimum height  $\sigma_{y_{\max}}$  and  $\sigma_{y_{\min}}$ , respectively. As we can put labels anywhere inside the traverse area, we can place up to  $\lceil \frac{\sigma_{x_{\max}}}{\sigma_{x_{\min}}} \rceil \cdot \lceil \frac{\sigma_{y_{\max}}}{\sigma_{y_{\min}}} \rceil$  labels intersecting that area and therefore  $\lceil \frac{\sigma_{x_{\max}}}{\sigma_{x_{\min}}} \rceil \cdot \lceil \frac{\sigma_{y_{\max}}}{\sigma_{y_{\min}}} \rceil$  overlaps occur during the movement (see Figure 12). This results in the following corollary, as an extension of Lemma 2.1.

► **Corollary A.1.** *When the labels are arbitrary rectangles with side length  $\sigma_{x_i}$  and  $\sigma_{y_i}$ , with  $1 \leq i \leq n$  and  $n$  denotes the number of labels, performing transition  $\mathcal{L}_1 \xrightarrow{RM_i A} \mathcal{L}_2$  for a label of a point  $p_i$  can result in at most  $\lceil \frac{\sigma_{x_{\max}}}{\sigma_{x_{\min}}} \rceil \cdot \lceil \frac{\sigma_{y_{\max}}}{\sigma_{y_{\min}}} \rceil$  overlaps given that the end position of  $p_i$  is free, where  $\sigma_{x_{\min}} = \min\{\sigma_{x_i} \mid 1 \leq i \leq n\}$  and  $\sigma_{x_{\max}}$ ,  $\sigma_{y_{\min}}$  and  $\sigma_{y_{\max}}$  are defined similarly.*

Corollary A.1 shows how upper bounds on the number of overlaps produced by square labels can be extended to the setting of arbitrary rectangles. This introduces only a constant factor, depending on the ratio between the largest and smallest side lengths in each dimension. However, for many transitions adding the constant factor, as suggested by Corollary A.1, does not yield a tight bound. This stems from the fact that many upper bounds require overlaps with the start or end position of a label  $l$ , not just the traversed area of  $l$ . Since



those positions are solely occupied at respectively the beginning and the end of the transition, we cannot place  $\lceil \frac{\sigma_{x_{\max}}}{\sigma_{x_{\min}}} \rceil \cdot \lceil \frac{\sigma_{y_{\max}}}{\sigma_{y_{\min}}} \rceil$  labels in those positions: many of those labels are unable to move away completely.



**Figure 12** The  $8 \times 8$  label wants to perform a counterclockwise diagonal movement and overlaps with nine stationary  $3 \times 3$  labels. The overlapping region is dotted, while the end position of the moving label is indicated with the dashed rectangle.

## B Detailed Case Study Results

In this case study we measure both the running times of our implementation, as well as objective metrics (overlaps and transition duration) of the computed transitions. In the next two sections, we outline these two measurements separately.

For this case study we used a standard laptop with the following specifications and software versions.

- Intel®Core™ i5-8265U CPU @ 1.60GHz with 16 Gigabyte RAM
- Windows 11 Pro 21H2 (64 Bit) and Microsoft Edge Browser 109.0.1518.78
- Neo4j 4.1.7, Java openjdk 11.0.2, vue 3.0.11, leaflet 1.7.1, and gsap 3.6.1
- External 27" monitor with a resolution of 1920×1080px

### B.1 Back-end computations

For the measurements of the back-end computations see Table 4.

### B.2 Transition measurements

For the results of our measurements on the computed transitions see Table 5.

■ **Table 4** Detailed results of the case study – Average running times in the back-end. QT = Query Time Database, CG CT = Conflict Graph Creation Time, MIS CT = MAXIMAL INDEPENDENT SET Computation Time.

Scenario	Setting	Conflict Graph (CG)				Changes in the Labeling				Running times [ms]			
		#Vertices	#Edges	#Added	#Removed	#Moved	QT	CG CT	MIS CT	Total			
Italy	1	68.00	384.20	5.80	4.40	2.00	478.53	7.40	6.00	579.24			
	2	110.16	635.49	1.51	0.86	1.14	213.62	3.79	4.25	302.12			
Lausanne	1	204.80	1,701.60	10.80	6.80	5.40	685.10	14.89	26.08	904.63			
	2	287.24	2,386.62	3.95	2.62	3.97	281.48	10.69	19.75	464.28			
Leeds	1	554.40	16,214.40	16.20	8.60	7.20	2,478.50	86.90	96.80	2,949.60			
	2	647.89	19,646.78	7.59	6.32	3.43	2,745.78	94.49	88.93	3,237.15			
Los Angeles	1	521.60	27,013.20	14.20	12.00	7.40	2,582.44	127.21	162.46	3,137.27			
	2	676.11	41,496.05	4.70	3.81	1.78	2,526.20	188.49	147.95	3,053.15			
New Delhi	1	1,490.40	268,126.80	21.20	12.00	10.00	9,764.81	1,505.77	1,197.61	12,645.40			
	2	1,714.16	309,493.78	6.81	5.35	2.59	7,154.48	1,385.80	1,010.94	9,732.02			
Sao Paolo	1	316.00	7,876.60	12.00	6.80	8.60	1,140.52	49.41	46.61	1,368.60			
	2	411.14	11,299.68	3.41	2.70	1.62	435.66	44.04	42.15	640.02			

■ **Table 5** Detailed results of the case study – Transition duration, number of overlaps, and time required to compute the transition for each transition style. #M = Number of moved labels, Trans. Comp. = Time required to compute the transition, Trans. Durat. = Transition duration.

Scenario	Setting	#M	Naive transitions						DAG-based transitions						Simultaneous transitions					
			#Overlaps			Trans. Comp. [ms]			Trans. Durat. [ms]			#Overlaps			Trans. Comp. [ms]			Trans. Durat. [ms]		
			Max	Avg.	Total	Max	Avg.	Total	Max	Avg.	Total	Max	Avg.	Total	Max	Avg.	Total	Max	Avg.	Total
Italy	1	10	2	0.40	2	7.20	2.66	6,499.60	2,200.86	1	0.20	1	9.60	3.38	4,488.30	1,398.12	3	6.80	2.88	2,503.00
	2	42	2	0.16	6	7.50	3.26	6,506.20	1,620.26	1	0.05	2	5.80	2.48	5,504.30	1,296.22	3	6.50	2.89	2,502.70
Lausanne	1	27	2	0.40	2	10.30	5.70	16,489.60	6,498.68	2	0.40	2	13.20	5.90	4,505.20	2,301.50	2	9.30	5.42	2,508.70
	2	147	7	1.43	53	14.80	7.02	18,496.60	5,273.01	6	0.78	29	14.50	7.41	9,493.80	3,238.79	7	1.19	5.79	2,504.40
Leeds	1	36	7	1.40	7	13.90	7.10	23,497.00	9,097.04	4	0.80	4	14.30	6.04	5,486.50	2,094.52	5	9.00	4.46	2,493.40
	2	127	6	1.03	38	11.10	6.46	16,499.50	4,810.41	5	0.59	22	12.40	5.94	7,503.30	2,917.80	4	0.65	4.92	2,507.70
Los Angeles	1	37	5	1.00	5	13.20	7.64	25,494.40	9,296.16	2	0.40	2	13.10	7.16	4,499.60	2,297.78	2	14.80	8.44	2,494.80
	2	66	4	0.43	16	8.50	4.70	8,505.80	2,553.35	4	0.30	11	11.00	5.29	4,501.50	1,877.12	3	0.38	5.18	2,502.70
New Delhi	1	50	3	1.20	6	17.80	9.90	22,481.00	10,796.62	2	0.60	3	15.80	9.28	8,487.50	4,595.52	4	1.40	7.78	2,502.90
	2	96	3	0.59	22	13.00	7.54	12,498.60	3,539.01	2	0.27	10	13.20	7.42	5,495.70	2,214.52	3	0.46	7.79	2,513.50
Sao Paolo	1	43	1	0.20	1	11.70	6.50	25,501.00	9,696.42	0	0.00	0	12.50	6.10	8,491.20	3,296.46	3	0.60	5.22	2,500.80
	2	60	5	0.41	15	10.40	4.86	13,496.00	2,281.82	4	0.24	9	8.20	5.11	8,491.20	1,688.23	4	0.38	4.62	2,503.20