Eurographics Conference on Visualization (EuroVis) 2022
R. Borgo, G. E. Marai, and T. Schreck
(Guest Editors)

Volume 41 (2022), Number 3

Branch Decomposition-Independent Edit Distances for Merge Trees Supplementary Material

Florian Wetzels, Heike Leitte, and Christoph Garth

Technische Universität Kaiserslautern

Appendix A: Unordered vs Ordered BDTs and other Relations

We illustrate the difference between the ordered distance from [SSW14] and the unordered distance from [PVDT21] on an example merge tree which can be seen in Figure 1. The three side branches b-f, c-g and d-h have a natural ordering defined by the scalar value of their saddles. This ordering is shown in the sibling orderings in the BDTs. Pont et al. allow arbitrary mappings, i.e. all three mappings are valid for their distance, whereas Saikia et al. only allow order-preserving mappings. This means that crossings in the mappings such as in the first one are not taken into account. This can also be phrased differently: The allowed mappings between child branches of a node correspond to string-edit mappings between the two child-sequences in this method by Saikia et al. whereas the method of Pont et al. checks for the optimal maximum matching between the two sets of children.

Furthermore, the unordered approach on BDTs used by Pont et al. gives them advantage over all methods that work directly on merge trees, as it allows for mappings that are ancestor-preserving in the BDT but not in the origial merge tree, as can be seen by the implicit mapping of the node with label b and c in Figure 1.

Relation of d_C, W_2^T and d_S We now discuss the relation of the three branch decomposition-dependent methods in Figure 4 in more detail, however, still in an intuitive manner without providing formal proofs for the claims. The constrained edit distance d_C is, as stated in the categorization, working on BDTs. In the original paper, it is defined to work on merge trees that are labeled with branch properties. This makes it equivalent (in terms of the mapping search space) to a constrained edit distance on *ordered* BDTs, since the ancestor preservation condition in merge trees translates to the ordering in BDTs. Therefore, the search space of d_C is a strict superset of the search space of d_S (as long as the same base metric is used).

As discussed before, the Wasserstein Distance on merge trees W_2^T uses d_1 as the underlying edit distance, as does d_S, however, it works on *unordered* BDTs. The lifting to unordered BDTs is the only difference (in terms of search space) between d_S and W_2^T. The only difference between d_S and d_C is the step from d_1 to d_c. As both variants require order preserving mappings, both consider strictly ordered BDTs. From these two observations, we can follow that

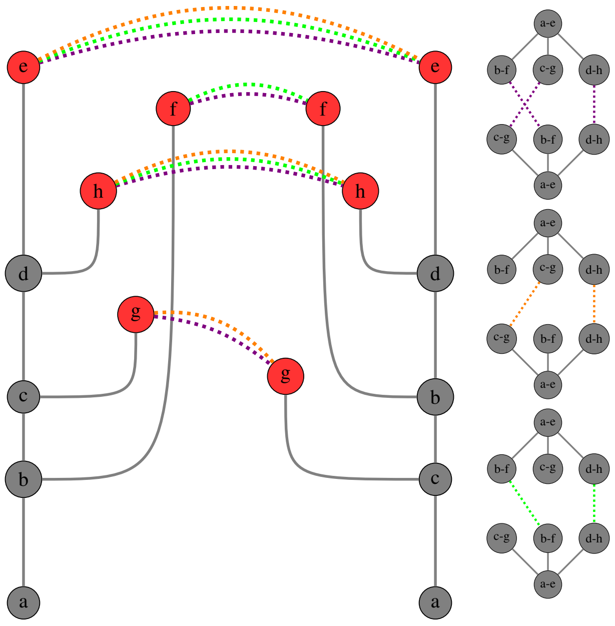

Figure 1: *On the left, we see a merge tree with three different mappings between the leaves in purple, green and orange. On the right, we see the three corresponding mappings between the branch decomposition graphs.*

the search space extensions of d_C and W_2^T in comparison to d_S are mutually exclusive and therefore d_S has to be the intersection of the two, as shown in Figure 4.

Appendix B: Proof of Theorem 1

For completeness, we first state the theorem again:

d_B is a metric on the set

$$\{(T,B) \mid T \text{ is an abstract merge tree}, B \in B(T)\},$$

as long as the cost function c on the branch labels is a metric.

Proof Given two trees T_1, T_2 with branch decompositions $B_1 \in$

$B(T_1), B_2 \in B(T_2)$, the first two metric properties, identity and symmetry are trivial, so only the triangle inequality remains to be proven. Therefore, consider a third tree T_3 with branch decomposition $B_3 \in B(T_3)$ and optimal mappings $M_{1,2} \subseteq B_1 \times B_2, M_{2,3} \subseteq B_2 \times B_3$ with $d_B(B_1,B_2) = \mathrm{c}(M_{1,2})$ and $d_B(B_2,B_3) = \mathrm{c}(M_{2,3})$. Now suppose that the triangle inequality is violated for the triple B_1,B_2,B_3, i.e. $d_B(B_1,B_3) > d_B(B_1,B_2) + d_B(B_2,B_3)$.

To show a contradiction, we construct another mapping $M_{1,3}$ between B_1 and B_3 in the following way. We define $(a,c) \in M_{1,3}$ for all branches $a \in B_1, c \in B_3$ where there is a branch $b \in B_2$ such that $(a,b) \in M_{1,2}$ and $(b,c) \in M_{2,3}$. Now consider the costs of the new mapping

$$\mathrm{c}(M_{1,3}) = \sum_{(a,c) \in \overline{M_{1,3}}} \mathrm{c}(a,c),$$

and specifically the single terms in the sum, pairs p with costs $\mathrm{c}(p) = \mathrm{c}(a,c)$. If $a \in B_1$ and $c \in B_3$, then there is a branch $b \in B_2$ such that $(a,b) \in M_{1,2}$ and $(b,c) \in M_{2,3}$. Then we know that $\mathrm{c}(a,b)$ and $\mathrm{c}(b,c)$ are terms in the sums of $\mathrm{c}(M_{1,2})$ and $\mathrm{c}(M_{2,3})$ and also that $\mathrm{c}(a,b) + \mathrm{c}(b,c) > \mathrm{c}(a,c) = \mathrm{c}(p)$ due to the metric property of the basic distance function. If $c \notin B_3$ (i.e. $c = \perp$), then either $(a,\perp) \in \overline{M_{1,2}}$ holds or $(a,b) \in \overline{M_{1,2}}$ and $(b,\perp) \in \overline{M_{2,3}}$. In both cases we know that $\mathrm{c}(a,\perp) \geq \mathrm{c}(p)$ and $\mathrm{c}(a,b) + \mathrm{c}(b,\perp) \geq \mathrm{c}(p)$. If $a \notin B_1$ (i.e. $a = \perp$), then either $(\perp,c) \in \overline{M_{2,3}}$ holds or $(\perp,b) \in \overline{M_{1,2}}$ and $(b,c) \in \overline{M_{2,3}}$. Again, in both cases we know that $\mathrm{c}(\perp,c) \geq \mathrm{c}(p)$ and $\mathrm{c}(a,b) + \mathrm{c}(b,\perp) \geq \mathrm{c}(p)$. In total, we can conclude that $\mathrm{c}(M_{1,3}) \leq \mathrm{c}(M_{1,2}) + \mathrm{c}(M_{2,3})$ and therefore $\mathrm{c}(M_{1,3}) \leq d_B(M_1,M_3)$, which leads to a contradiction. □

Appendix C: Proof of Lemma 2

For completeness, we state the lemma again:

Given two merge trees T_1,T_2 with roots v_1,u_1, let v_2,u_2 be the unique children of the two roots and let those have children v_3,v_4 and u_3,u_4. Let T_1' be the subtree rooted in (v_2,v_3), T_1'' rooted in (v_2,v_4), T_2' rooted in (u_2,u_3) and T_2'' in (u_2,u_4). Let M be an optimal branch mapping for T_1 and T_2. Then, for the optimal cost of M it holds that:

- $d_B(T_1,T_2) = d_B(T_1',\perp) + d_B(T_1 - T_1',T_2)$ or
- $d_B(T_1,T_2) = d_B(\perp,T_2') + d_B(T_1,T_2 - T_2')$ or
- $d_B(T_1,T_2) = d_B(T_1'',\perp) + d_B(T_1 - T_1'',T_2)$ or
- $d_B(T_1,T_2) = d_B(\perp,T_2'') + d_B(T_1,T_2 - T_2'')$ or
- $d_B(T_1,T_2) = d_B(T_1',T_2') + d_B(T_1 - T_1',T_2 - T_2')$ or
- $d_B(T_1,T_2) = d_B(T_1'',T_2'') + d_B(T_1 - T_1'',T_2 - T_2'')$ or
- $d_B(T_1,T_2) = d_B(T_1',T_2'') + d_B(T_1 - T_1',T_2 - T_2'')$ or
- $d_B(T_1,T_2) = d_B(T_1'',T_2') + d_B(T_1 - T_1'',T_2 - T_2')$

Proof To prove this lemma, we will do two nested case distinctions. First, consider the optimal mapping M. It is build upon two branch decompositions $B_1 \in B(T_1)$ and $B_2 \in B(T_2)$. Since the roots have degree one, in both branch decompositions the main branch has to go through v_2 and u_2. Therefore, in B_1 either the edge (v_2,v_3) or the edge (v_2,v_4) is contained in the main branch, and in B_2 either (u_2,u_3) or (u_2,u_4).

Let us first consider the case that (v_2,v_4) and (u_2,u_4) are part of the main branches of B_1 and B_2. Then there is a branch $a = v_2v_3...v_l \in B_1$ and a branch $b = u_2u_3...u_l \in B_2$ and we know that $B_1[T_1']$ and $B_2[T_2']$ exist. For the mapping M we have the following options:

(A) $(a,b) \in M$: In this case the subtrees T_1' and T_2' are mapped to each other, i.e. $M[B_1[T_1']] = M[B_2[T_2']]$ and it holds that $\mathrm{c}(M) = \mathrm{c}(M[B_1[T_1']]) + \mathrm{c}(M[B_1[T_1 - T_1']])$ which then means that $M[B_1[T_1']]$ and $M[B_1[T_1 - T_1']]$ are optimal mappings between T_1' and T_2' and between $T_1 - T_1'$ and $T_2 - T_2'$, as otherwise they could be replaced in M by better mappings contradicting the optimality of M.

(B) $(a,b') \in M$, but $b' \neq b$ is not a branch of T_2': If a is mapped to a branch other than b this means that b and all its descendant branches are not mapped in M as this would either contradict the order condition or the parent condition of branch mappings (conditions 3 and 4, Definition 2).

(C) $(a',b) \in M$, but $a' \neq a$ is not a branch of T_1': If b is mapped to a branch other than a this means that a and all its descendant branches are not mapped in M as this would either contradict the order condition or the parent condition of branch mappings (conditions 3 and 4, Definition 2).

(D) There is no branch a' of T_1' and no branch b' of T_2 with $(a,b') \in M$ or $(a',b') \in M$.

In each case, we can conclude the following for M and $\mathrm{c}(M) = d_B(T_1,T_2)$:

(A) Since $M[B_1[T_1']]$ and $M[B_1[T_1 - T_1']]$ are optimal, we can conclude that

$$d_B(T_1,T_2) = d_B(T_1',T_2') + d_B(T_1 - T_1',T_2 - T_2').$$

(B) Since T_2' is not covered by M, we can conclude that it is mapped to $\perp$ in $\overline{M}$ and therefore

$$d_B(T_1,T_2) = d_B(\perp,T_2') + d_B(T_1,T_2 - T_2').$$

(C) Since T_1' is not covered by M, we can conclude that it is mapped to $\perp$ in $\overline{M}$ and therefore

$$d_B(T_1,T_2) = d_B(T_1',\perp) + d_B(T_1 - T_1',T_2).$$

(D) Since T_1' and T_2' are not covered by M, we can conclude that they are mapped to $\perp$ in $\overline{M}$ and therefore

$$d_B(T_1,T_2) = d_B(T_1',\perp) + d_B(T_1 - T_1',T_2)$$

and

$$d_B(T_1,T_2) = d_B(\perp,T_2') + d_B(T_1,T_2 - T_2')$$

both hold.

Together with the other cases for B_1 and B_2, which yield analogous equations, we get all the terms from above. □

Appendix D: Proof of Lemmma 3

For completeness, we state the lemma again:

Given a merge tree T_1 with root v_1, let v_2 be the unique children of the root and let it have children v_3,v_4. Let T_1' be the subtree rooted in (v_2,v_3) and T_1'' rooted in (v_2,v_4). Then, for the optimal cost of $M_\perp(T_1)$ it holds that:

- $d_B(T_1,\perp) = d_B(T_1',\perp) + d_B(T_1 - T_1',\perp)$ or
- $d_B(T_1,\perp) = d_B(T_1'',\perp) + d_B(T_1 - T_1'',\perp)$,

and $d_B(\perp,T_2)$ decomposes symmetrically.

Proof If one of the trees is empty, then we only have to find the branch decomposition for the other one that has the lowest cost. Locally, we only have two cases. Either v_3 lies on the main branch or v_4 does. In either case, the optimal branch decomposition $B_1 \in B(T_1)$ decomposes into those of the trees listed in the lemma, as seen in Section 3.1. If v_3 is on the main branch of B_1, then

$$\mathrm{c}(B_1,\perp) = \mathrm{c}(B_1[T_1''],\perp) + \mathrm{c}(B_1[T_1 - T_1''],\perp)$$

and if v_4 is on the main branch of B_1, then

$$\mathrm{c}(B_1,\perp) = \mathrm{c}(B_1[T_1'],\perp) + \mathrm{c}(B_1[T_1 - T_1'],\perp). \quad \square$$

Appendix E: Proof of Lemmma 4

For completeness, we state the lemma again:

Given two merge trees $T_1 = (\{v_1,v_2\},\{(v_2,v_1)\})$, $T_2 = (\{u_1,u_2\},\{(u_2,u_1)\})$ that only have one branch, the following holds for d_B:

- $d_B(T_1,\perp) = \mathrm{c}(v_1v_2,\perp)$,
- $d_B(\perp,T_2) = \mathrm{c}(\perp,u_1u_2)$ and
- $d_B(T_1,T_2) = \mathrm{c}(v_1v_2,u_1u_2)$.

Proof For $d_B(T_1,\perp)$ and $d_B(\perp,T_2)$ this follows directly from the definition of $M_\perp(T_1), M_\perp(T_2)$ and the uniqueness of the branch decompositions.
For $d_B(T_1,T_2)$ it follows directly from the requirement to map the main branches (condition 2 in Definiton 2) and the uniqueness of the branch decompositions. $\square$

Appendix F: Algorithm

We now give the pseudo code for an algorithm computing the distance d_B. Again, we only show the recursion for binary trees, but it is easy to adapt for trees of arbitrary degree (see below). Algorithm 1 shows the recursive procedure without memoization.

Theorem 1 Let T_1,T_2 be abstract merge trees with roots r_1,r_2 and let r_1',r_2' be their unique child. Then $d_B(r_1,r_1',r_2,r_2')$ in Algorithm 1 computes the branch mapping distance between T_1 and T_2.

Proof Algorithm 1 strictly resembles the recursion from Lemmas 2-4, where T_1 is represented by (n_1,p_1), T_1' by $(c_{1,1},n_1)$, $T_1 - T_1'$ by $(c_{1,1},p_1)$, T_1'' by $(c_{1,2},n_1)$, $T_1 - T_1'$ by $(c_{1,2},p_1)$ and $T_2,T_2',T_2'',T_2 - T_2',T_2 - T_2''$ symmetrically. This correspondence then also yields the correctness of the initial call $d_B(r_1,r_1',r_2,r_2')$ for T_1,T_2. $\square$

Adaption for Arbitrary Degree. For the purpose of readability, we omitted a detailed description of the algorithm for unbounded degree, but we now give a short discussion on which adaptations have to be made. To adapt the algorithm for trees of arbitrary degree, we have to replace the last four options in line 23 through an optimal matching of the subtrees, similar to the algorithm for the constrained edit distance by Zhang [Zha96]. For each pair of possible main branches in the two trees, we have to find the optimal matching between the remaining children. For the constrained edit distance, this adds a factor of $(\deg_1 + \deg_2) \cdot \log(\deg_1 + \deg_2)$, but since we still have to find the optimal pair of main branches, we get another factor of $\deg_1 \cdot \deg_2$. To obtain a more precise upper bound, one would have to study how these costs amortize over the whole tree, since there are obvious limits to the sum of degrees (e.g. there can only be a constant number of degree $\Omega(n)$ nodes), but we omit this analysis as merge trees of high degree do usually not appear in realistic scenarios.

Algorithm 1: Computing the branch mapping distance

```
1  Function $d_B(n_1,p_1,n_2,p_2)$:
2      if $n_1 = \perp$ and $n_2$ is a leaf then
3          return $\mathrm{c}(\perp, p_2...n_2)$
4      if $n_2 = \perp$ and $n_1$ is a leaf then
5          return $\mathrm{c}(p_1...n_1, \perp)$
6      if $n_1$ is a leaf and $n_2$ is a leaf then
7          return $\mathrm{c}(p_1...n_1, p_2...n_2)$
8      if $n_1 = \perp$ and $n_2$ is an inner node then
9          Let $c_{2,1}, c_{2,2}$ be the children of $n_2$
10         return min $\begin{cases} d_B(\perp,\perp,c_{2,1},p_2) + d_B(\perp,\perp,c_{2,2},n_2) \\ d_B(\perp,\perp,c_{2,2},p_2) + d_B(\perp,\perp,c_{2,1},n_2) \end{cases}$
11     if $n_2 = \perp$ and $n_1$ is an inner node then
12         Let $c_{1,1}, c_{1,2}$ be the children of $n_1$
13         return min $\begin{cases} d_B(c_{1,1},p_1,\perp,\perp) + d_B(c_{1,2},n_1,\perp,\perp) \\ d_B(c_{1,2},p_1,\perp,\perp) + d_B(c_{1,1},n_1,\perp,\perp) \end{cases}$
14     if $n_1$ is a leaf and $n_2$ is an inner node then
15         Let $c_{2,1}, c_{2,2}$ be the children of $n_2$
16         return min $\begin{cases} d_B(n_1,p_1,c_{2,1},p_2) + d_B(\perp,\perp,c_{2,2},n_2) \\ d_B(n_1,p_1,c_{2,2},p_2) + d_B(\perp,\perp,c_{2,1},n_2) \end{cases}$
17     if $n_2$ is a leaf and $n_1$ is an inner node then
18         Let $c_{1,1}, c_{1,2}$ be the children of $n_1$
19         return min $\begin{cases} d_B(c_{1,1},p_1,n_2,p_2) + d_B(c_{1,2},n_1,\perp,\perp) \\ d_B(c_{1,2},p_1,n_2,p_2) + d_B(c_{1,1},n_1,\perp,\perp) \end{cases}$
20     if $n_1$ is an inner node and $n_2$ is an inner node then
21         Let $c_{1,1}, c_{1,2}$ be the children of $n_1$
22         Let $c_{2,1}, c_{2,2}$ be the children of $n_2$
23         return min $\begin{cases} d_B(c_{1,1},p_1,n_2,p_2) + d_B(c_{1,2},n_1,\perp,\perp) \\ d_B(c_{1,2},p_1,n_2,p_2) + d_B(c_{1,1},n_1,\perp,\perp) \\ d_B(n_1,p_1,c_{2,1},p_2) + d_B(\perp,\perp,c_{2,2},n_2) \\ d_B(n_1,p_1,c_{2,2},p_2) + d_B(\perp,\perp,c_{2,1},n_2) \\ d_B(c_{1,1},p_1,c_{2,1},p_2) + d_B(c_{1,2},n_1,c_{2,2},n_2) \\ d_B(c_{1,1},n_1,c_{2,1},n_2) + d_B(c_{1,2},p_1,c_{2,2},p_2) \\ d_B(c_{1,2},p_1,c_{2,1},p_2) + d_B(c_{1,1},n_1,c_{2,2},n_2) \\ d_B(c_{1,2},n_1,c_{2,1},n_2) + d_B(c_{1,1},p_1,c_{2,2},p_2) \end{cases}$
```

Appendix G: Experimental Runtimes

Using a C++ implementation of Algorithm 1, we perform benchmarks on the datasets from Section 5. We use different simplification thresholds for the heated cylinder dataset, the synthetic outlier ensemble, and the vortex street dataset. For the latter two, we first add artificial noise. The observed running times can be seen in Table 1. For each dataset, we compute the branch mapping dis-

	HC (10)	O (20)	VS (68)	VS (135)	O (196)	VS (213)	HC (233)	VS (259)	O (356)
Runtime	$1.5 \cdot 10^{-5}$s	$9.5 \cdot 10^{-5}$s	$3 \cdot 10^{-3}$s	$7 \cdot 10^{-2}$s	$3.7 \cdot 10^{0}$s	$1.1 \cdot 10^{0}$s	$6.5 \cdot 10^{0}$s	$1.5 \cdot 10^{0}$s	$4.2 \cdot 10^{1}$s

Table 1: *Running times of the branch mapping distance on the datasets from Section 5: The synthetic outlier dataset (O), the heated cylinder (HC) and the vortex strees (VS). We used different versions of the datasets with different levels of noise and simplification. The sizes of the merge trees are shown in brackets.*

tance for 100 random pairs. The table shows the average sizes of the merge trees as well as the average running times over all 100 pairs. We expect that further optimization of our code (e.g. parallelization) could yield practical running times for even larger trees. We leave this for future work.

Appendix H: Applications

We now show the more detailed results on some of the datasets from Section 5. We start with renderings of the full synthetic ensembles in Figures 3 and 4. Furthermore, in these figures the clustermaps are shown together with dendograms of the underlying hierarchical clusterings.

To show that this example is independent of the chosen parameters (Wasserstein distance and squared costs), we provide the distance matrices for the branch mappings and constrained edit mappings also for the L_∞- and overhang cost from [SMKN20] in Figure 2. Furthermore, Figure 5 shows another ensemble and the distance matrices of the branch mapping and constrained edit distance, where we did not use the squared costs. We were unable to do these extra comparisons for the Wasserstein distance, as its TTK-implementation does not allow for a base metric change.

Figure 6 shows the complete distance matrix for the dataset from Section 5.2 and Figure 7 a direct comparison with our implementation of the distance by Sridharamurthy et al. [SMKN20] on the same dataset.

In Figure 12, we provide a comparison of our tracking results on the ion density dataset with the results of the Wasserstein distance on the same dataset provided by Pont et al. in [PVDT21].

We also provide the distance matrices for noisier versions of the outlier ensemble and the vortex street dataset. For the outlier example, we did this on a noisy version of the ensemble from Figure 5 and a new outlier ensemble that was directly created with noise. Figures 10 and 11 show that the outlier is still visible if we add artificial noise to the synthetic ensemble, however, not as clearly. The same holds for the periodic pattern in the vortex street dataset with artificial noise, which can be seen in Figures 9 and 8. Here, the difference to the normal version is more significant. The periodicity detection is possible, but clearly impeded. We should note that an improved robustness against this kind of noise is not the goal of our approach, but rather robustness against *structural* perturbations like in the original synthetic ensemble, which was its motivation. We included these examples with noise to show that our method has *some* robustness agaist this type of noise, comparable to other merge tree-based methods, but it is not significantly improved. In fact, it might even perform worse in some cases due to the generally smaller distances.

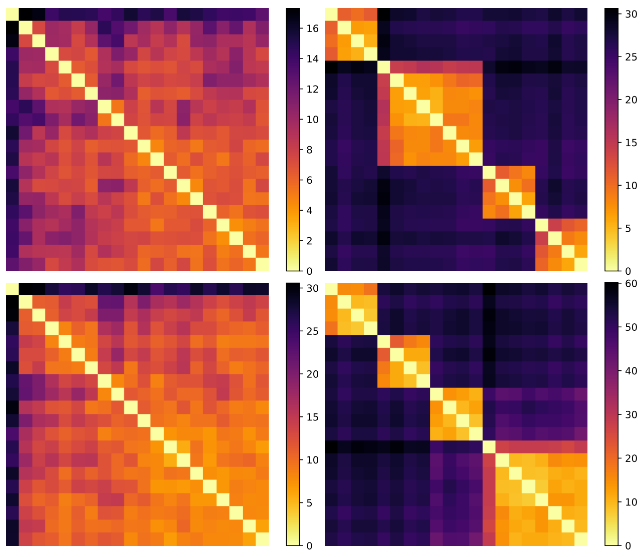

Figure 2: *Comparison of branch mapping distance and constrained edit distance on the outlier ensemble using different base metrics. The distance matrices for the branch mapping distance can be seen on the left and for the constrained edit distance on the right. The top matrices represent the distances using the L_∞ base metric and the bottom matrices the distances using the Overhang base metric.*

References

[PVDT21] PONT M., VIDAL J., DELON J., TIERNY J.: Wasserstein distances, geodesics and barycenters of merge trees. *IEEE Transactions on Visualization and Computer Graphics* (2021), 1–1. doi:10.1109/TVCG.2021.3114839. 1, 4, 11

[SMKN20] SRIDHARAMURTHY R., MASOOD T. B., KAMAKSHIDASAN A., NATARAJAN V.: Edit distance between merge trees. *IEEE Trans. Vis. Comput. Graph. 26*, 3 (2020), 1518–1531. doi:10.1109/TVCG.2018.2873612. 4, 8

[SSW14] SAIKIA H., SEIDEL H., WEINKAUF T.: Extended branch decomposition graphs: Structural comparison of scalar data. *Comput. Graph. Forum 33*, 3 (2014), 41–50. doi:10.1111/cgf.12360. 1

[Zha96] ZHANG K.: A constrained edit distance between unordered labeled trees. *Algorithmica 15*, 3 (1996), 205–222. doi:10.1007/BF01975866. 3

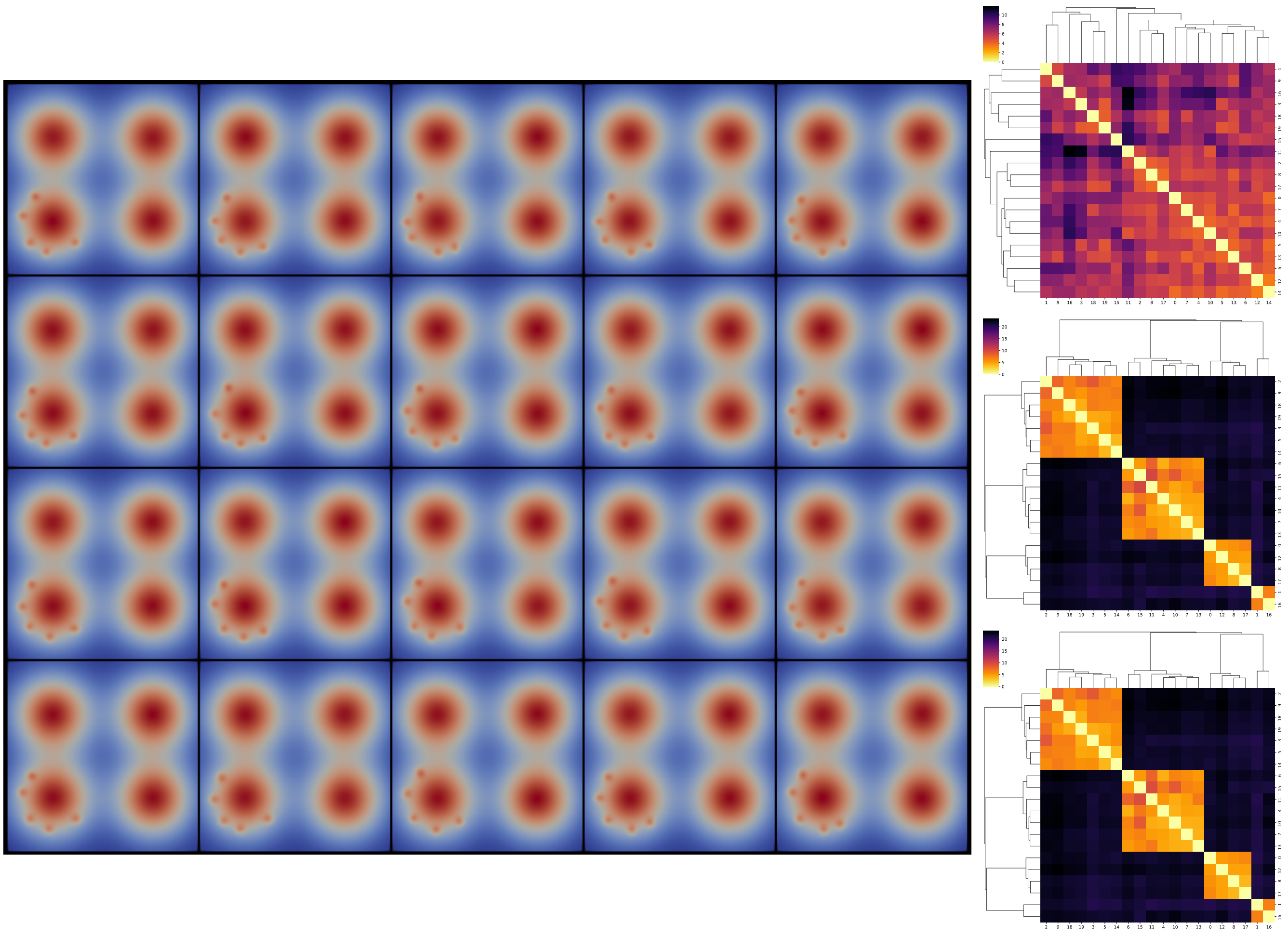

Figure 3: *The example ensemble (left) with a comparison of branch mapping distance (top right), constrained edit distance (center right) and Wasserstein distance (bottom right). Heatsmaps for distance matrices of the three distance measures are shown where the rows and columns are ordered by the clustermap function of the seaborn library to clearly identify clusters.*

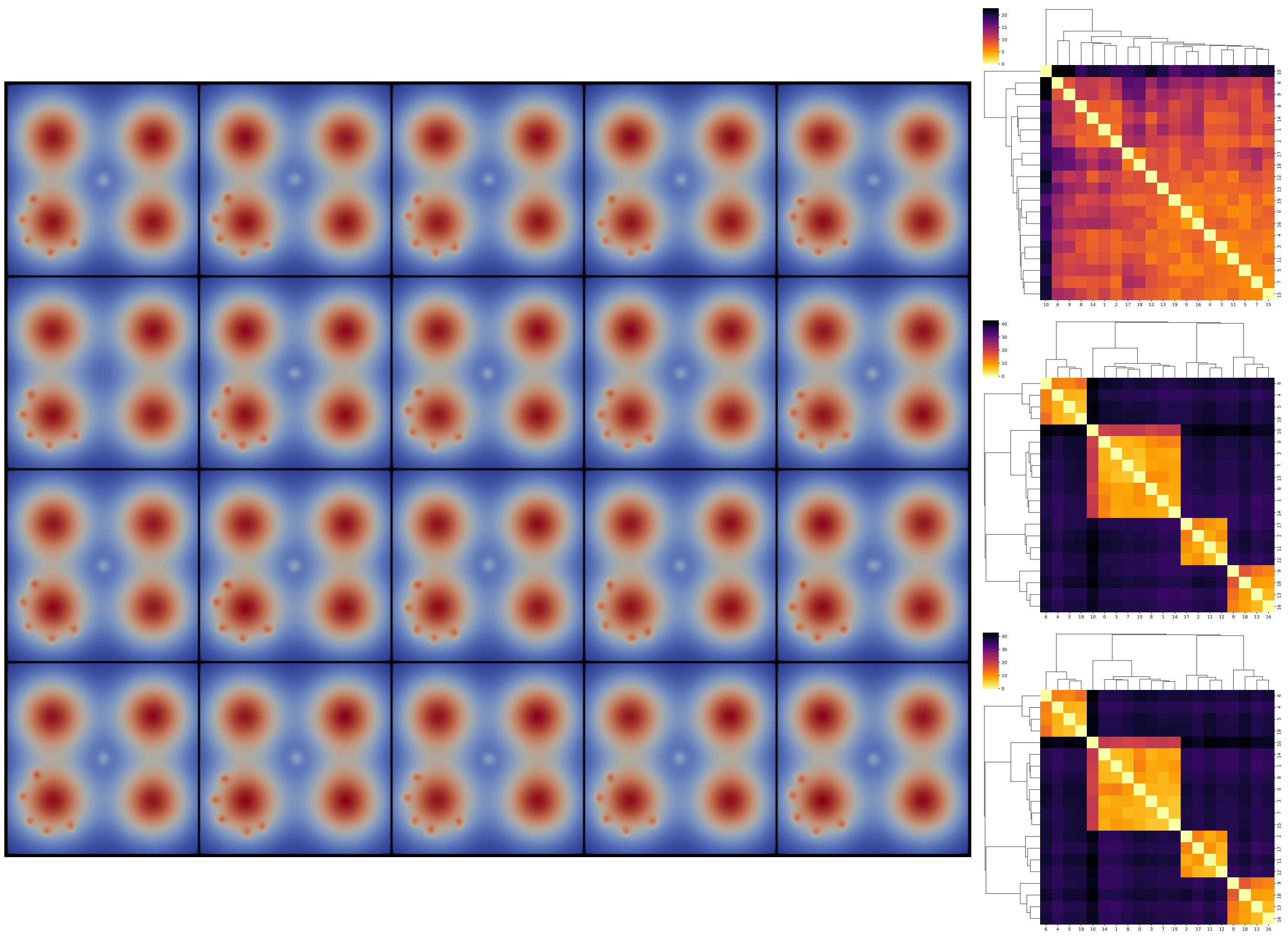

Figure 4: *The outlier ensemble (left) with a comparison of branch mapping distance (top right), constrained edit distance (center right) and Wasserstein distance (bottom right). Heatsmaps for distance matrices of the three distance measures are shown where the rows and columns are ordered by the clustermap function of the seaborn library to clearly identify clusters. The plots also show the dendograms of the underlying clusterings.*

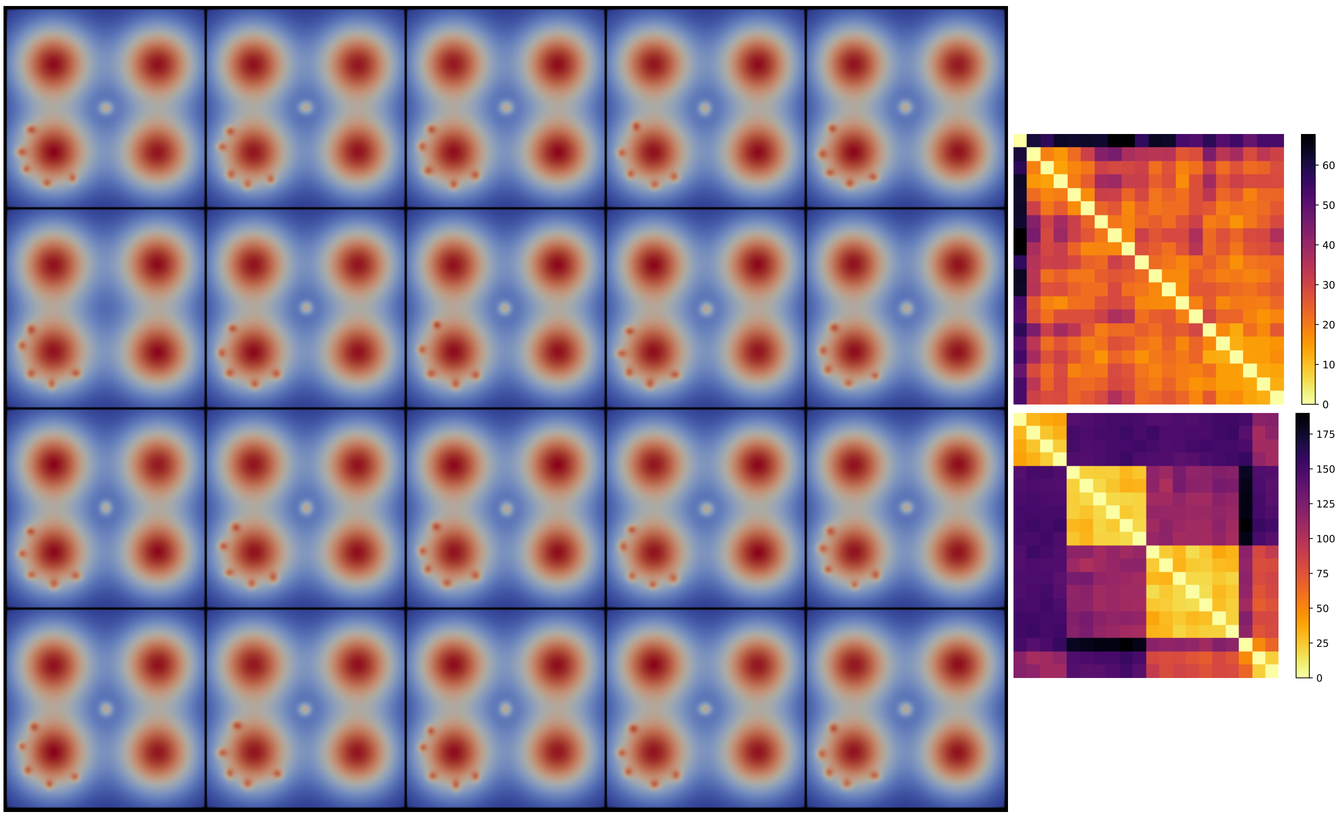

Figure 5: *Another outlier ensemble (left) with a comparison of branch mapping distance (top right) and constrained edit distance (bottom right). Heatmaps for distance matrices of the two distance measures are shown where the rows and columns are ordered by the clustermap function of the Seaborn library to clearly identify clusters.*

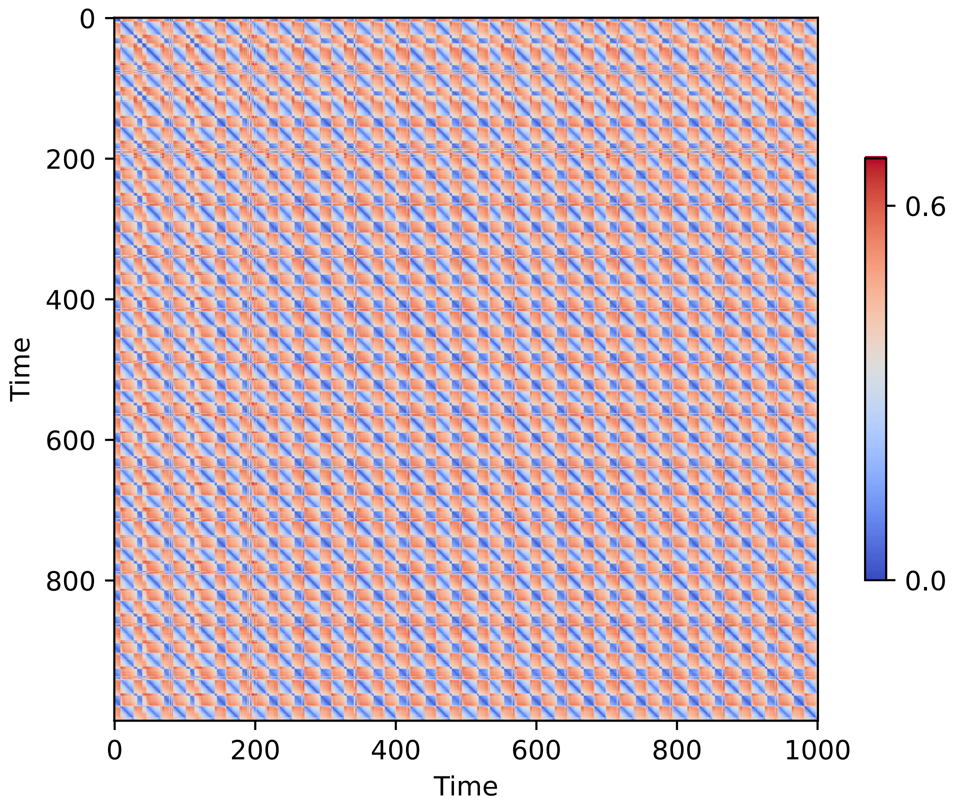

Figure 6: *The complete distance matrix for all 1001 time steps of the vortex street dataset.*

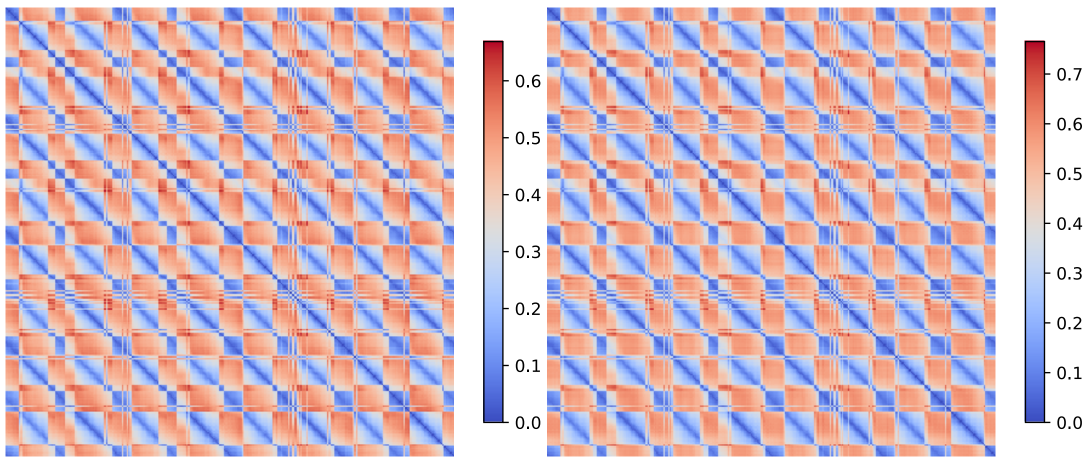

Figure 7: *Comparison of the (partial) distance matrices of the vortex street dataset using the branch mapping distance (left) and the constrained edit distance (right). The right image was computed using our own implementation of the constrained edit distance and resembles Figure 13 in [SMKN20] (small differences can be due to different merge tree computation or different ways of attaching branch properties to vertices). Both, or better to say all three, matrices show the same periodic pattern in the data.*

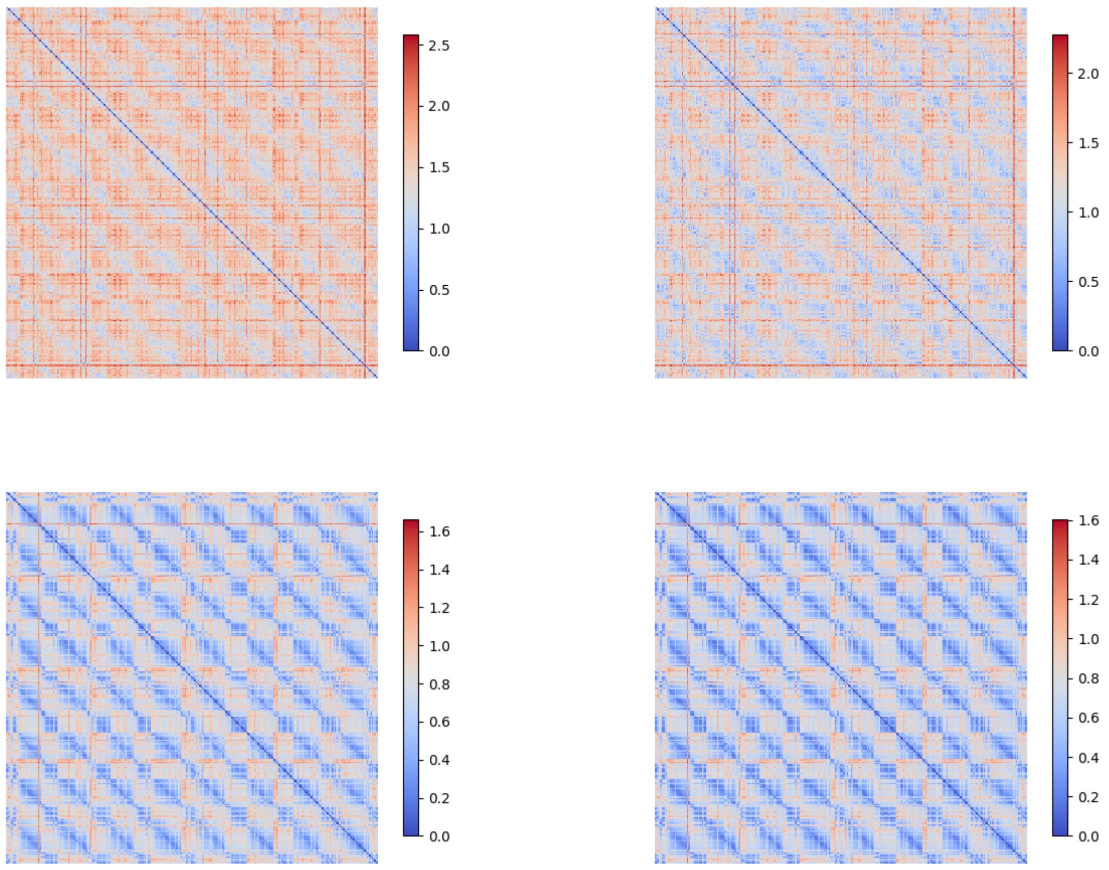

Figure 8: *Distance matrices (using Wasserstein distance as base metric) for noisy versions of the vortex street dataset. Four different variants of noise are shown using noise of different intensity/amplitude (top* 1%*, bottom* 0.4% *of scalar range) and different simplification (left* ≈ 200 *vertices, right* ≈ 150 *vertices, original data* ≈ 65 *vertices).*

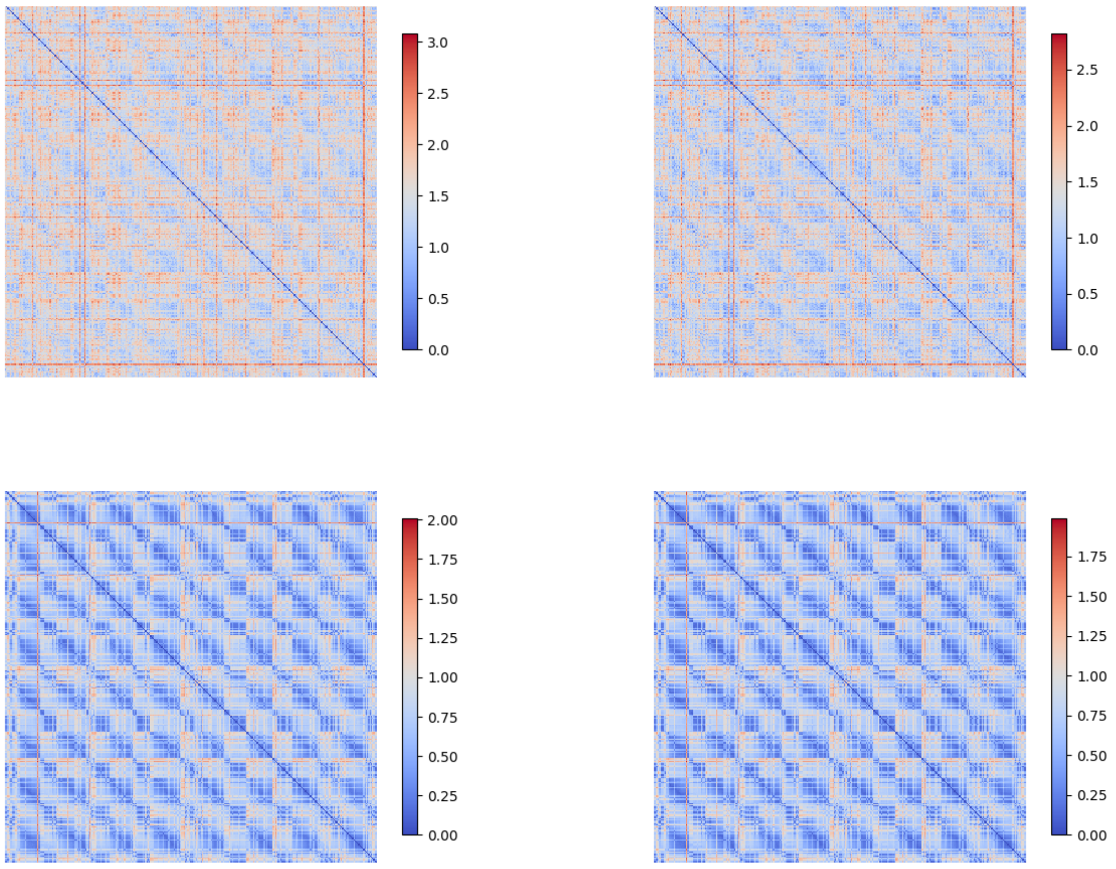

Figure 9: *Distance matrices (using persistence difference as base metric) for noisy versions of the vortex street dataset. Four different variants of noise are shown using noise of different intensity/amplitude (top* 1%*, bottom* 0.4% *of scalar range) and different simplification (left* ≈ 200 *vertices, right* ≈ 150 *vertices, original data* ≈ 65 *vertices).*

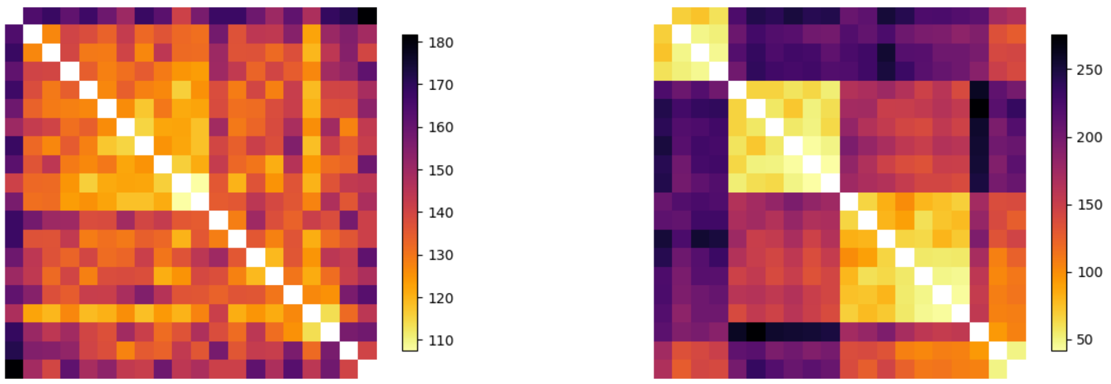

Figure 10: *Distance matrices for noisy versions of the outlier ensemble from Figure 5. The left matrix was computed using the branch mapping distance, the right one using the constrained edit distance. The merge trees were simplified to* ≈ 200 *vertices.*

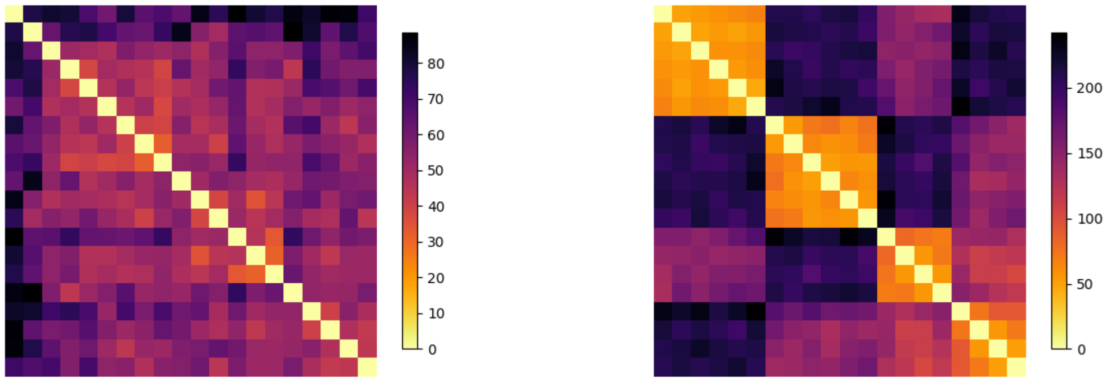

Figure 11: *Distance matrices for a new outlier ensemble with noise. The left matrix was computed using the branch mapping distance, the right one using the constrained edit distance. The merge trees were simplified to ≈ 200 vertices.*

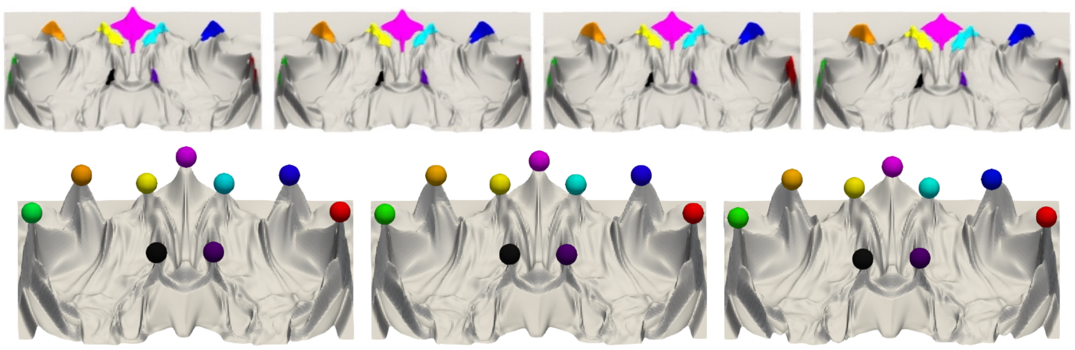

Figure 12: *Comparison of the tracking on the ion density dataset using the branch mapping distance (top) and the Wasserstein distance (bottom). For the bottom image, we replicated the mapping from Figure 11 in [PVDT21] using the TTK implementation of the Wasserstein distance. It is easy to see that both methods yield the same semantically meaningful matching.*