



Deposited via The University of Sheffield.

White Rose Research Online URL for this paper:

<https://eprints.whiterose.ac.uk/id/eprint/176406/>

Version: Supplemental Material

Article:

Magallanes, J., Stone, T., Paul D, M. et al. (2022) Sequen-C: A multilevel overview of temporal event sequences. *IEEE Transactions on Visualization and Computer Graphics*, 28 (1). pp. 901-911. ISSN: 1077-2626

<https://doi.org/10.1109/TVCG.2021.3114868>

© 2021 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other users, including reprinting/ republishing this material for advertising or promotional purposes, creating new collective works for resale or redistribution to servers or lists, or reuse of any copyrighted components of this work in other works. Reproduced in accordance with the publisher's self-archiving policy.

Reuse

Items deposited in White Rose Research Online are protected by copyright, with all rights reserved unless indicated otherwise. They may be downloaded and/or printed for private study, or other acts as permitted by national copyright laws. The publisher or other rights holders may allow further reproduction and re-use of the full text version. This is indicated by the licence information on the White Rose Research Online record for the item.

Takedown

If you consider content in White Rose Research Online to be in breach of UK law, please notify us by emailing eprints@whiterose.ac.uk including the URL of the record and the reason for the withdrawal request.

Sequen-C: A Multilevel Overview of Temporal Event Sequences (Supplemental Materials)

Jessica Magallanes, Tony Stone, Paul D Morris, Suzanne Mason, Steven Wood, and Maria-Cruz Villa-Uriol

This document presents further details regarding the time performance analysis (Appendix A), Sequen-C implementation details (Appendix B), and discusses domain-specific examples for the analytic tasks (Appendix C).

APPENDIX A. TIME PERFORMANCE ANALYSIS

This appendix adds further details on the time complexity, and presents additional tables and charts of the performance analysis of different data subsets. The time performances reported in this paper were obtained in a laptop with the following characteristics:

- macOS Big Sur Version 11.4
- MacBook Pro (15-inch, 2017)
- Processor 2.9 GHz Quad-Core Intel Core i7
- Memory 16 GB
- Graphics Radeon Pro 560 4GB

Time complexity

The time complexity of Algorithm 1 is $O(Nnl^2)$, where N is the number of input unique sequences, l is the maximum sequence length in the dataset, and n is the average number of sequences per node. Note that l is the maximum sequence length in the dataset, as opposed to the average sequence length, because the length of a new alignment $n_{a \cup b}$ will be at least the maximum sequence length amongst the sequences in the nodes n_a and n_b . If a sequence with a high length is aggregated at an early iteration, this length will be carried to all subsequent alignments. As observed in Fig. 10 and Fig. 11, the alignment time increases significantly when the maximum sequence length in the dataset increases.

The function `aggregate` is repeated $N - 1$ times, from which the alignment step (MSA) is the most time consuming with a complexity of $O(nl^2)$. Fig. 8 and Fig. 9 show plots comparing the running time in seconds for the functions in Algorithm 1 (`buildAggregateTree`) and Algorithm 2 (`Align`, `Score and Simplify`), according to the maximum sequence length of the four subsets from the CUREd and MIMIC-III data shown in Table 1 (in the paper). Note that these plots should show a quadratic curve given the term l^2 in the time complexity, however, they look quite linear due to the huge gap between the first three maximum lengths and the last one (e.g. 13, 19, 28, and 177 for CUREd).

Performance for additional subsets of the data

Table 1 in the paper shows the time performance for subsets of 25%, 50%, 75%, and 100% of the data. Tables 2 and 3 show the time for subsets increasing every 5%. Only the time for `buildAggregateTree` and `Align` are included as these are the ones that are precomputed (see Appendix B for more details). Unique sequences were ordered by length before selecting subsets, meaning that the 5% subset contains the shortest sequence. The additional subsets allow us to see how the maximum sequence length impacts the time performance, even when the number of unique sequences does not increase much between subsets - for example when comparing the alignment time between the 95% and 100% data subsets (see Fig. 10 and Fig. 11).

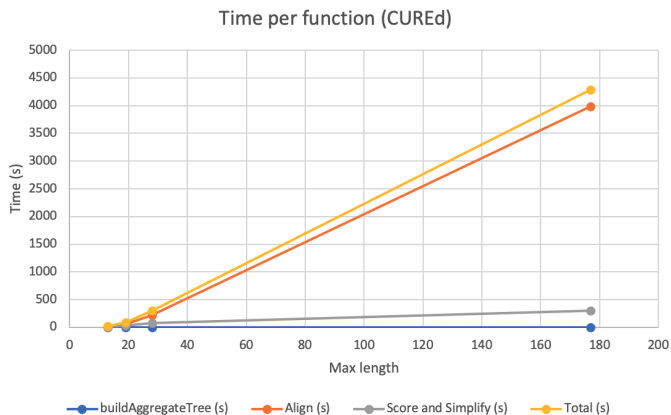


Fig. 8: Running time per function, according to the maximum sequence length (13, 19, 28, and 177) of four subsets of the CUREd data. The `Align` function is the most time consuming.

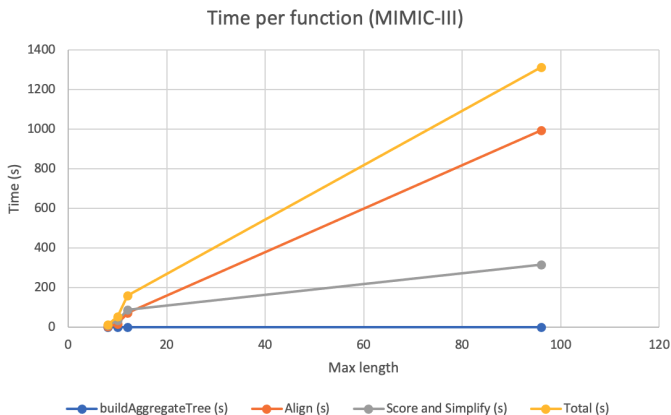


Fig. 9: Running time per function, according to the maximum sequence length (8, 10, 12, and 96) of four subsets of the MIMIC-III data. The `Align` function is the most time consuming.

APPENDIX B. IMPLEMENTATION DETAILS

The GUI of Sequen-C was implemented in Java, while the clustering and alignment steps were implemented in R.

As mentioned in Appendix A, a big factor in the performance is the maximum sequence length in the dataset (l). Around 300 to 500 input unique sequences with an average sequence length of 7 to 10 events can be aligned relatively fast (under 10 seconds), as long as the length of certain sequences do not go too far from the average. For Sequen-C to have real time interaction, Algorithms 1 and 2 are precomputed, except for the `Score and Simplify` steps which are computed on the fly as the value of the number of clusters (k) or information score threshold (I_τ) change.

The R script precomputes the aggregate tree and the alignment of each node in the tree. The hierarchy of clusters and alignment matrices are saved to a file. For a selected number of clusters k , the alignment

| CURED dataset | | | | | | | |
|---------------|---------------|--------|-----------------|---------------------|---------|--------------------|---------|
| Subset | No. sequences | | No. event types | Length of sequences | | Execution time (s) | |
| | Individual | Unique | | Average | Maximum | buildAggregateTree | Align |
| 5% | 14232 | 48 | 11 | 7.33 | 9 | 0.01 | 0.38 |
| 10% | 17869 | 96 | 11 | 8.43 | 10 | 0.00 | 1.23 |
| 15% | 19065 | 144 | 11 | 8.96 | 11 | 0.01 | 3.17 |
| 20% | 19696 | 192 | 11 | 9.47 | 11 | 0.02 | 5.70 |
| 25% | 20124 | 240 | 11 | 9.97 | 13 | 0.03 | 9.27 |
| 30% | 20200 | 289 | 11 | 10.56 | 14 | 0.04 | 13.49 |
| 35% | 20417 | 337 | 11 | 11.15 | 15 | 0.05 | 19.29 |
| 40% | 20738 | 385 | 11 | 11.76 | 17 | 0.06 | 32.71 |
| 45% | 20794 | 433 | 11 | 12.44 | 18 | 0.17 | 41.27 |
| 50% | 20901 | 481 | 11 | 13.07 | 19 | 0.10 | 59.11 |
| 55% | 21006 | 529 | 11 | 13.69 | 20 | 0.12 | 83.32 |
| 60% | 21327 | 577 | 11 | 14.28 | 22 | 0.14 | 105.54 |
| 65% | 21400 | 625 | 11 | 14.87 | 23 | 0.17 | 143.82 |
| 70% | 21461 | 673 | 11 | 15.52 | 25 | 0.19 | 174.42 |
| 75% | 21520 | 722 | 11 | 16.25 | 28 | 0.21 | 225.50 |
| 80% | 21583 | 770 | 11 | 17.04 | 30 | 0.27 | 297.74 |
| 85% | 21657 | 818 | 11 | 17.88 | 33 | 0.26 | 421.03 |
| 90% | 21707 | 866 | 11 | 18.88 | 38 | 0.47 | 614.18 |
| 95% | 21757 | 914 | 11 | 20.07 | 48 | 0.36 | 895.94 |
| 100% | 21805 | 962 | 11 | 22.70 | 177 | 0.37 | 3984.67 |

Table 2: Time performance of the buildAggregateTree and Align functions for additional subsets of the CURED data.

| MIMIC-III dataset | | | | | | | |
|-------------------|---------------|--------|-----------------|---------------------|---------|--------------------|--------|
| Subset | No. sequences | | No. event types | Length of sequences | | Execution time (s) | |
| | Individual | Unique | | Average | Maximum | buildAggregateTree | Align |
| 5% | 135 | 66 | 51 | 4.42 | 5.00 | 0.05 | 0.19 |
| 10% | 240 | 131 | 88 | 4.94 | 6.00 | 0.01 | 0.40 |
| 15% | 310 | 197 | 112 | 5.37 | 7.00 | 0.12 | 1.49 |
| 20% | 375 | 262 | 139 | 5.77 | 7.00 | 0.04 | 2.31 |
| 25% | 442 | 328 | 158 | 6.07 | 8.00 | 0.07 | 3.46 |
| 30% | 507 | 393 | 184 | 6.39 | 8.00 | 0.10 | 6.00 |
| 35% | 573 | 459 | 199 | 6.62 | 8.00 | 0.20 | 6.44 |
| 40% | 638 | 524 | 221 | 6.88 | 9.00 | 0.17 | 8.74 |
| 45% | 704 | 590 | 231 | 7.11 | 9.00 | 0.21 | 10.63 |
| 50% | 770 | 656 | 242 | 7.31 | 10.00 | 0.22 | 16.91 |
| 55% | 835 | 721 | 261 | 7.55 | 10.00 | 0.35 | 24.75 |
| 60% | 901 | 787 | 277 | 7.76 | 10.00 | 0.34 | 33.90 |
| 65% | 966 | 852 | 301 | 7.99 | 11.00 | 0.36 | 37.57 |
| 70% | 1032 | 918 | 322 | 8.21 | 11.00 | 0.48 | 48.80 |
| 75% | 1097 | 983 | 338 | 8.45 | 12.00 | 0.51 | 73.36 |
| 80% | 1163 | 1049 | 355 | 8.71 | 13.00 | 1.30 | 77.56 |
| 85% | 1228 | 1114 | 373 | 8.99 | 14.00 | 0.80 | 109.02 |
| 90% | 1294 | 1180 | 388 | 9.34 | 16.00 | 1.22 | 147.31 |
| 95% | 1359 | 1245 | 412 | 9.79 | 20.00 | 1.37 | 228.06 |
| 100% | 1425 | 1311 | 448 | 10.68 | 96.00 | 1.46 | 995.39 |

Table 3: Time performance of the buildAggregateTree and Align functions for additional subsets of the MIMIC-III data.

matrices of the k clusters are retrieved from the file by the Java program. Then, the Score and Simplify steps are computed on the fly over the retrieved alignment matrices according to the current value of I_τ . The times reported in Table 1 (in the paper) refer to the time required to compute Score and Simplify for all the nodes in the tree, however, in the implementation these steps are computed only for the selected values of k or I_τ . On average, it takes 0.11 seconds to recompute Score/Simplify and repaint the visualization of a given number of clusters, this allows users to change the vertical and horizontal level-of-detail in real time. Depending on the dataset, the total preprocessing time can be obtained as the sum of the buildAggregateTree and Align columns on Tables 2 and 3.

APPENDIX C. DOMAIN-SPECIFIC EXAMPLES

The analytic tasks used to design Sequen-C were defined through a series of face-to-face (pre-COVID) and online interviews, and feedback sessions with three groups of stakeholders in the clinical domain (emergency services, cardiac intensive care, and outpatients). With the exception of the expert in the cardiac intensive care domain, experts were not only *problem owners*, but they were also *data owners*, and they had an active interest in understanding their own data to improve how they operated.

Data characterization

The event sequence data owned by the stakeholders were partially derived from *processes*. This means that for *some* of the captured events, there was an internal ordering of events. For example, an

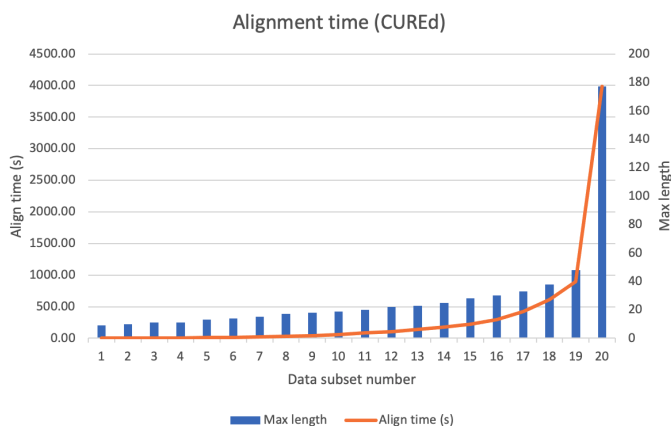


Fig. 10: Alignment time in seconds and maximum sequence length for the 20 different data subsets shown in Table 2 for the CUREd dataset.

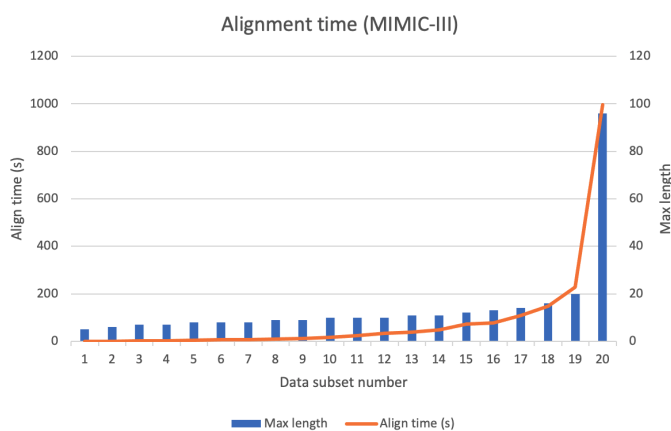


Fig. 11: Alignment time in seconds and maximum sequence length for the 20 different data subsets shown in Table 3 for the MIMIC-III dataset.

ambulance does not arrive at the scene unless a call requesting such service happens first, or a patient cannot see a consultant before waiting to see that consultant.

In the case of the CUREd dataset, some events come in groups, such as “Patient arrived Emergency Department”, “Triage”, and “Patient seen for treatment”. These events are semantically related, they usually go together and in this order. If they do not, they are considered to be a deviating pathway worth exploring, possibly indicating a glitch in the data capturing system. For these three events, there could be other events in between such as “Ambulance stood down”, an event related to the ambulance transporting the patient to the Emergency Department. The interest of the domain experts was to obtain frequent and infrequent pathways originated from the calls, identify patients with multiple incidents, and explore the aggregated characteristics of the patients following a pathway of interest. For the MIMIC-III case study, the interest was in understanding the impact that certain combinations of prescribed drugs (irrespective of the administration order) had on the clinical outcome for a patient (e.g. death or length of hospital stay). In these cases, enforcing the strict ordering of events when grouping patients in clusters of similar patients was not the best strategy.

Domain-specific examples

Below we provide domain-specific examples for the analytic tasks listed in Section 4.

T1. Explore common and deviating pathways: help users to explore and discover which clusterings summarize better the most common (and deviating) pathways in the data. Clusters will group

sequences that share a set of event types, regardless of their order. Examples:

- “When studying the evolution of patients over time and the influence of certain prescriptions, we would like to obtain a summary of the distinct clinical pathways, by grouping patients that have been prescribed with similar drug combinations (not necessarily in the same order).”
- “We would like to identify and explore patients with unique drug combinations that differ from the rest of patients.”
- “When studying patients calling emergency services, we would like to obtain a summary of the distinct pathways originated from the calls according to the occurrence or absence of certain events in our process. For example, by grouping patients according to whether they attended the emergency department or not, or whether they required an ambulance or not, or whether their call was just handled via a phone call.”
- “We would like to identify what proportion of patients have an unusual number of calls to the emergency department and explore their outcomes.”

T2. Interpret the sequences that constitute a cluster: the visualization should allow users to compare the most common event orderings (and permutations) within and across clusters using sequence alignment. Examples:

- “When studying patients calling emergency services, we would like to quickly understand what events usually occur after a visit to the emergency department.”
- “We would like to identify patients that make multiple calls before an ambulance arrives at the scene, and compare them with patients that receive an ambulance service after the first call.”
- “We would like to identify scenarios that *do not make sense* from a process perspective and that are likely to be data input errors, to improve how data is captured in practice. For example, study scenarios where it appears the patient has been seen by a health professional before a call has been made to the emergency department.”
- “When studying the evolution of patients over time and the influence of certain prescriptions, we would like to understand how the length of stay varies depending on the drug combination prescribed to patients across care units.”

T3. Focus the analysis on a selected set of records: allow queries in the dataset to focus on sequences with specific characteristics. Examples:

- “We would like to focus our analyses on patients above 70 years old with a diagnosis of heart disease.”
- “For the patients calling emergency services, we would like to focus our analyses on those patients who required an ambulance and ended up being admitted.”

T4. Obtain details on demand: provide coordinated views so that users can request finer details of interesting items in the overview. Users should be able to go from the highest level of aggregation (i.e. clusters), passing through sequences grouped by their unique sequence, to individual sequences and their raw data including event timestamps and duration. Examples:

- “We would like to explore the characteristics of individual patients within a cluster of interest - including the time elapsed between a call and the arrival of an ambulance at the scene, or their total length of stay.”

T5. Aggregate and compare context information for selected groups of records: the system should allow to aggregate and compare data attributes (e.g. age, gender, country) for selected clusters, unique sequences, or individual sequences. Examples:

- “We would like to understand the demographics of interesting patient groups, for example, patients whose call do not lead to an ambulance service or patients with a recorded death event.”

Influence of event ordering

The utility of Sequen-C in analyzing temporal event sequences and choice of distance metric, depends on the nature of the dataset and the questions from the end users. The proposed approach works well in situations where there are common events across sequences that act as milestone events, or in situations when certain events are semantically associated and will have a natural order between them. However, the current choice of distance metric will not work, or a different distance metric will be necessary, in cases where the interest is in studying the impact of event order in the outcome. For example:

- For patients in cardiac arrest, does the order of certain diagnostic tests impact patient outcome (e.g. length of stay, death)? Does the length of stay reduce or increase when an echocardiogram is performed before an angiogram?
- Following on the last domain-specific example for Task T2, when we want to study the evolution of patients over time and the influence of certain prescriptions, besides exploring drug combinations, does the order of medication impact the final patient outcome (e.g. medicine A prescribed before medicine B)?

In the cases above, when event order is an important criteria to partition patients into groups, another metric such as the Levenshtein edit distance could be used. Further work is necessary to test the utility of the proposed multilevel overview with alternative metrics that take into account order.