Mining Frequent Parallel Episodes with Selective Participation

Christian Borgelt[1] **Christian Braune**[2] **Kristian Loewe**[2,3] **Rudolf Kruse**[2]

[1]Intelligent Data Analysis Research Unit, European Centre for Soft Computing
c/ Gonzalo Gutiérrez Quirós s/n, 33600 Mieres (Asturias), Spain
[2]Department of Knowledge and Language Processing, Otto-von-Guericke-University
Universitätsplatz 2, 39106 Magdeburg, Germany
[3]Department of Neurology, Experimental Neurology, Otto-von-Guericke-University
Leipziger Straße 44, 39120 Magdeburg, Germany

Abstract

We consider the task of finding frequent parallel episodes in parallel point processes, allowing for imprecise synchrony of the events constituting occurrences (temporal imprecision) as well as incomplete occurrences (selective participation). We tackle this problem with frequent pattern mining based on the CoCoNAD methodology, which is designed to take care of temporal imprecision. To cope with selective participation, we form a reduction sequence of items (event types) based on found frequent patterns and guided by pattern overlap. We evaluate the performance of our method on a large number of data sets with injected parallel episodes.

Keywords: parallel episode, temporal imprecision, selective participation, frequent pattern mining

1. Introduction

We present methodology to identify *frequent patterns* in parallel point processes, a task that is also known as finding *frequent parallel episodes* in event sequences (see [10]). This task can be seen as a generalization of frequent item set mining (FIM, see e.g. [1]). However, while in FIM items occur in predefined groups, so-called *transactions*, in our setting a continuous (time) scale underlies the data.

Continuous time already poses a first problem, which we refer to as *temporal imprecision*: in practice, synchrony of events is almost never perfect, but affected by a certain amount of (temporal) jitter. We take this into account when mining for frequent patterns by defining that items (or events) co-occur if they occur in a (user-defined) limited time span from each other. As a support measure we use the size of a maximum independent set (MIS) of such synchronous groups of items, which can be computed efficiently with a greedy algorithm [2, 12].

A second problem consists in *selective participation*: due to imperfections of the measuring technology or properties of the underlying process, several occurrences of a pattern may be incomplete, that is, only a subset of the items underlying the pattern is actually present in an instance. This problem is the focus of this paper and we tackle it with an approach that interprets found (partial) patterns as edges in a hypergraph. We then try to find a dense subgraph with a reduction sequence approach loosely inspired by the procedure in [15]: the total set of items is reduced by removing in each step the item that is least connected to the other items. The step in which the least connected item is most strongly connected (compared to all other steps) finally identifies the frequent pattern (or parallel episode).

The application domain that motivates our investigation is the analysis of *parallel spike trains* in neurobiology: sequences of points in time, one per neuron, representing the times at which an electrical impulse (*action potential* or *spike*) is emitted. Our objective is to identify *neuronal assemblies*, intuitively understood as groups of neurons that tend to exhibit synchronous spiking. Such cell assemblies were proposed in [6] as a model for encoding and processing information in biological neural networks. In particular, as a (possible) first step in the identification of neuronal assemblies, we look for *frequent neuronal patterns* (i.e. groups of neurons that exhibit *frequent synchronous spiking*). In this setting, both temporal imprecision and selective participation (of neurons) are expected to be present and thus require proper treatment.

The remainder of this paper is structured as follows: Section 2 covers basic terminology and notation. Sections 3 and 4 briefly review the CoCoNAD methodology for finding frequent parallel episodes in the presence of temporal imprecision. In Section 5 we present our methodology to identify frequent parallel episodes with selective participation. Section 6 reports experimental results on data sets with injected parallel episodes. Finally, in Section 7 we draw conclusions from our discussion.

2. Event Sequences and Parallel Episodes

We mainly adopt here the notation and terminology of [10]. That is, our data are (finite) *sequences of events* of the form $\mathcal{S} = \{\langle i_1, t_1 \rangle, \ldots, \langle i_m, t_m \rangle\}$, $m \in \mathbb{N}$, where i_k in the *event* $\langle i_k, t_k \rangle$ is the *event type* or *item* (taken from an item base B)

and $t_k \in \mathbb{R}$ is the time of occurrence of i_k, $k \in \{1, \dots, m\}$. Note that the fact that $\mathcal{S}$ is a set implies that there cannot be two events with the same item occurring at the same time: events with the same item must differ in their occurrence time and events occurring at the same time must have different types/items. Note also that such data may as well be represented as *parallel point processes* $\mathcal{P} = \{\langle i_1, \{t_1^{(1)}, \dots, t_{m_1}^{(1)}\}\rangle, \dots, \langle i_n, \{t_1^{(n)}, \dots, t_{m_n}^{(n)}\}\rangle\}$ by grouping events with the same item $i \in B$, $n = |B|$, and listing the times of their occurrences for each of them. Finally, note that in our motivating application (i.e. spike train analysis), the items (or event types) are the neurons and the corresponding point processes list the times at which spikes were recorded for these neurons.

Episodes (in $\mathcal{S}$) are defined as sets of items $I \subseteq B$ that are endowed with a partial order and usually required to occur in $\mathcal{S}$ within a certain time span. *Parallel episodes*, on which we focus in this paper, have no constraints on the relative order of their elements. An *instance (or occurrence) of a parallel episode* $I \subseteq B$, $I \neq \emptyset$, (or a *(set of) synchronous event(s)* for I) in an event sequence $\mathcal{S}$ with respect to a (user-specified) time span $w \in \mathbb{R}^+$ can be defined as a subsequence $\mathcal{R} \subseteq \mathcal{S}$, which contains exactly one event per item $i \in I$ and which can be covered by a (time) window at most w wide. Hence the set of all instances of a parallel episode $I \subseteq B$, $I \neq \emptyset$, in an event sequence $\mathcal{S}$ is

$$\begin{aligned}\mathcal{E}_{\mathcal{S},w}(I) \;=\; \big\{&\mathcal{R} \subseteq \mathcal{S} \mid \{i \mid \langle i,t\rangle \in \mathcal{R}\} = I \\ &\wedge |\mathcal{R}| = |I| \wedge \sigma_w(\mathcal{R}) > 0\big\},\end{aligned}$$

where the operator σ_w captures the (approximate) synchrony of the events in $\mathcal{R}$:

$$\sigma_w(\mathcal{R}) = \begin{cases} 1 & \text{if } \max\{t \mid \langle i,t\rangle \in \mathcal{R}\} \\ & \quad - \min\{t \mid \langle i,t\rangle \in \mathcal{R}\} \leq w, \\ 0 & \text{otherwise.}\end{cases}$$

That is, $\sigma_w(\mathcal{R}) = 1$ iff all events in $\mathcal{R}$ can be covered by a (time) window at most w wide.

Based on this notion of (imprecise) synchrony, we define the support of an item set $I \subseteq B$ as follows (see also [8, 13] for a related, but still significantly different characterization that is based on covering windows rather than sets of underlying events):

$$\begin{aligned}&s_{\mathcal{S},w}(I) = \\ &\quad \max\big\{|\mathcal{U}| \;\mid\; \mathcal{U} \subseteq \mathcal{E}_{\mathcal{S},w}(I) \;\wedge \\ &\qquad\quad \forall \mathcal{R}_1, \mathcal{R}_2 \in \mathcal{U}; \mathcal{R}_1 \neq \mathcal{R}_2 : \mathcal{R}_1 \cap \mathcal{R}_2 = \emptyset\big\}.\end{aligned}$$

That is, we define the support (or total synchrony) of a pattern $I \subseteq B$ as the size of a maximum independent set (MIS) of its instances (where by *independent set* we mean a collection of instances that do not share any events, that is, the instances do not overlap). Such an approach has the advantage that the resulting support measure is guaranteed to be anti-monotone, as can be shown generally for maximum independent subset (or, in a graph interpretation, node set) approaches—see, e.g., [4] or [16].

A parallel episode $I \subseteq B$ is called *frequent* (in $\mathcal{S}$) if its support $s_{\mathcal{S},w}(I)$ meets or exceeds a (user-specified) minimum support $s_{\min}$. The task of mining frequent parallel episodes consists in finding, for a given event sequence $\mathcal{S}$ and window width w, all parallel episodes $I \subseteq B$ that are frequent in $\mathcal{S}$. However, in order to reduce the output, it is common to report only the *closed frequent parallel episodes*, where a parallel episode I is called *closed* if no parallel episode that is a proper superset $J \supset I$ has the same support. We denote the set of all closed frequent parallel episodes that can be found in an event sequence $\mathcal{S}$ w.r.t. (user-specified) window width w and minimum support s_{min} by $\mathcal{C}_{\mathcal{S}}(w, s_{\min}) \subseteq 2^B$.

3. CoCoNAD

At least at first sight, a support measure based on (the size of) a maximum independent set (MIS) seems to suffer from the severe drawback that in the general case finding a maximum independent set is NP-complete [7] and even hard to approximate [5]. Intuitively speaking, this means that (unless $\mathsf{P} = \mathsf{NP}$) there is no (known) algorithm that does fundamentally better than an algorithm that tries all possibilities. As a consequence, the algorithm has exponential time complexity (in the size of the set $\mathcal{E}_{\mathcal{S},w}(I)$, from which the maximum independent set is to be selected) and thus would take a prohibitively long time to find a solution.

Fortunately, though, the problem instances we are facing here are strongly constrained by the underlying one-dimensional time domain, which makes it possible to devise an efficient greedy algorithm that solves it exactly. For a given item set I, for which the support is to be determined, this algorithm starts with an empty selection of instances and proceeds by traversing the sequence $\mathcal{S}$ (or the parallel point processes $\mathcal{P}$) chronologically. It always selects as the next instance the element of $\mathcal{E}_{\mathcal{S},w}(I)$ that does not overlap any of the already selected instances and contains the earliest possible events for each of the items in I. For this, it does not even have to construct the set $\mathcal{E}_{\mathcal{S},w}(I)$ explicitly, but can work directly on the sequence $\mathcal{S}$ (or the parallel point processes $\mathcal{P}$). As a consequence, it has a time complexity of $m_I \cdot \log(|I|)$, where $m_I = \sum_{i \in I} m_i$ is the sum of the numbers of events of each item i (that is, the total number of events with items in I), since m_I events have to be passed through a priority queue of size $|I|$. Details of this algorithm (including pseudo-code) can be found in [2], while a proof that it is guaranteed to find (the size of) a maximum independent set of $\mathcal{E}_{\mathcal{S},w}(I)$ can be found in [12].

Based on this support computation, frequent parallel episodes are then found with a standard divide-and-conquer scheme as it is also known from frequent item set mining, particularly from the Eclat algorithm [17, 1]. The algorithm proceeds as follows: for a chosen item i, the problem of finding all frequent parallel episodes is split into two subproblems: (1) find all frequent parallel episodes containing i and (2) find all frequent parallel episodes *not* containing i. Each subproblem is then further divided based on another item j: find all frequent patterns containing (1.1) both i and j, (1.2) i but not j, (2.1) j but not i, (2.2) neither i nor j etc.

The search is pruned with the so-called *apriori property*, which is a direct consequence of the fact that support is anti-monotone: $\forall I, J \subseteq B : (J \supseteq I \wedge s_{\mathcal{S},w}(I) < s_{\min}) \Rightarrow s_{\mathcal{S},w}(J) < s_{\min}$. Or in words: *no superset of an infrequent parallel episode can be frequent.* Hence, the recursive division process can be terminated as soon as the support of the set of all included split items falls below the (user-specified) minimum support $s_{\min}$. Details of this approach in the context of FIM can be found, for example, in [1]. Details of this scheme for finding (closed) frequent parallel episodes (including pseudo-code) can be found in [2]. Referring to the application domain that motivated its development, this algorithm is called CoCoNAD (for **Co**ntinuous-time **C**l**o**sed **N**euron **A**ssembly **D**etection).

4. Pattern Spectrum Filtering

A common problem of frequent pattern mining methods is the usually huge number of patterns they produce, most of which are merely chance events. In order to cope with this problem, [11] introduced and [14] refined an approach that is referred to as *pattern spectrum filtering* (PSF) and that tries to reduce the patterns to statistically significant ones.

This method is based on the following insight: even if it is highly unlikely that a *specific group* of z items co-occurs s times, it may still be likely that *some group* of z items co-occurs s times, even if items occur independently. The reason is simply that there are so many possible groups of z items (unless the item base B as well as the group size z are tiny) that even though each group has only a tiny probability of co-occurring s times, it may be almost certain that *one of them* co-occurs s times.

Therefore, since there is no *a-priori* reason to prefer certain sets of z items over others (although a refined analysis may take individual item frequencies into account), we should not declare a found pattern significant if the occurrence of a counterpart (same or larger group size z or support s) can be explained as a chance event under the null hypothesis of independent items. Hence, patterns with the same *pattern signature* $\langle z, s\rangle$ are pooled. This corresponds to conducting only one hypothesis test per pattern signature, either accepting or rejecting *all* patterns with this signature in a single decision. The main advantage of this approach is that it reduces the number of statistical tests considerably and thus offers an effective way of dealing with the *multiple testing problem*, which would be insurmountable if all patterns had to be tested individually.

In order to determine the likelihood of observing different pattern signatures $\langle z, s\rangle$ under the null hypothesis of independent items, a data randomization or surrogate data approach is employed. The general idea is to represent the null hypothesis implicitly by (surrogate) data sets that are generated from the original data in such a way that their occurrence probability is (approximately) equal to their occurrence probability under the null hypothesis. Such an approach has the advantage that it needs no explicit data model for the null hypothesis, which in many cases (including the one we are dealing with here) may be difficult to specify. Instead, the original data is modified in random ways to obtain data that are at least analogous to those that could be sampled under conditions in which the null hypothesis holds. An overview of surrogate data methods in the context of neural spike train analysis can be found in [9].

We employ pattern spectrum filtering to remove (many) patterns that are merely chance events, in order to simplify forming a reduction sequence of the items (see Section 5): the fewer patterns we have to handle, the faster we can generate this sequence and the less affected it will be by chance events. However, we apply pattern spectrum filtering in a less strict form than [11, 14], because many of the patterns that result from parallel episodes with incomplete occurrences may be relatively small or have a relatively low support and thus are likely to be removed by strict pattern spectrum filtering. How strict pattern spectrum filtering is, is mainly controlled by the number of surrogate data sets: the more surrogate data sets are generated, the more patterns will be identified as chance events.

Furthermore, in order to avoid having to actually generate and analyze surrogate data sets (which is a time consuming process), we draw on the approach proposed in [3], which estimates a pattern spectrum by analyzing the original data set. This approach has the additional advantage that it reduces fluctuations in case only few surrogate data sets are to be generated (as is the case here, because we want to execute only weak pattern spectrum filtering), since it estimates a support distribution per pattern size.

5. Selective Participation

Our approach to identify parallel episodes in the presence of selective participation is based on the following insight: although incomplete occurrences of a pattern may make it impossible that the full pattern is reported by the mining procedure, it is highly likely that several overlapping subsets will be reported. This is illustrated in Figure 1, which shows parallel spike trains of six neurons a to f with complete and incomplete instances of the parallel episode comprising all six neurons (in blue; while background spikes are shown in gray). Although the full set of neurons fires together only once (left-

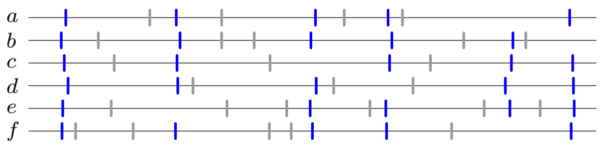

Figure 1: Parallel episodes (indicating neuron assembly activity) with selective participation (blue) as well as background spikes (gray).

most instance) and thus would not be detected, the other five incomplete occurrences give rise to five subsets of size 4, each of which occurs twice, and many subsets of size 3, occurring 3 or more times. Since these patterns overlap heavily, it should be possible to reconstruct the full pattern by analyzing pattern overlap and combining patterns.

Our method views the set of patterns that have been found in a given data set as a *hypergraph*[1] on the set of items (which are the vertices of this hypergraph): each pattern forms a hyperedge. Patterns that are affected by selective participation thus give rise to densely connected sub-hypergraphs. Hence, we should be able to identify such patterns by finding densely connected sub-hypergraphs.

For detecting dense sub-hypergraphs we draw on the approach proposed in [15]. Although this approach was designed to find dense subgraphs in standard graphs, its basic idea is easily transferred and adapted: we form a reduction sequence of items by removing, in each step, the item that is least connected to the other items (that are still considered). Then we identify from this sequence the set of items where the least connected item (i.e., the one that was removed next) was most strongly connected (compared to other steps of the sequence). This item set is the result of the procedure.

Although this limits the basic procedure to the identification of a single pattern, it is clear that multiple patterns can easily be found with the same amendment as suggested in [15]: find a pattern and then remove the underlying items (vertices) from the data. Repeat the procedure on the remaining items to find a second pattern. Remove the items of this second pattern and so on. A drawback of this approach is that it can find only disjoint patterns and thus fails to identify overlapping patterns. However, given the general difficulty to handle selective participation, we believe that this is an acceptable shortcoming for the time being.

Formally, we construct a reduction sequence of item sets, starting from the item base B, as

$$\begin{aligned} J_n &= B, \qquad \text{where } n = |B|, \\ J_k &= J_{k+1} - \{\text{argmin}_{i \in J_{k+1}} \xi_{\mathcal{S},w,s_{\min}}(i, J_{k+1})\}, \\ &\quad \text{for } k = n-1, n-2, \ldots, 0, \end{aligned}$$

where $\xi_{\mathcal{S},w,s_{\min}}(i, J_k)$ denotes the strength of connection that item $i \in J_k$ has to the other items in the set J_k, as it is induced by the (closed) frequent patterns found when mining the sequence $\mathcal{S}$ with window width w and minimum support $s_{\min}$ (concrete functions $\xi_{\mathcal{S},w,s_{\min}}(i, J_k)$ are studied below). We then assign a quality measure to each element of this reduction sequence:

$$\begin{aligned} &\forall k; 0 \leq k \leq n : \\ &\qquad \xi_{\mathcal{S},w,s_{\min}}(J_k) = \min_{i \in J_k} \xi_{\mathcal{S},w,s_{\min}}(i, J_k). \end{aligned}$$

Finally, we select the pattern (item set)

$$I = \text{argmax}_{J_k; 0 \leq k \leq n} \xi_{\mathcal{S},w,s_{\min}}(J_k),$$

that is, the pattern with the highest quality (sub-hypergraph density), as the result of our procedure.

To obtain concrete instances of the functions $\xi_{\mathcal{S},w,s_{\min}}(i, J_k)$ we tried two different approaches:

Pattern-based approach
Let $\mathcal{C}^*_{\mathcal{S}}(w, s_{\min}) \subseteq 2^B$ be the set of closed frequent patterns that are identified by the CoCoNAD algorithm (if executed with window width w and minimum support $s_{\min}$ on $\mathcal{S}$), for which no counterpart (same pattern signature) was observed in any of the surrogate data sets (that is, the closed frequent patterns remaining after pattern spectrum filtering). Let $\mathcal{C}^*_{\mathcal{S},J}(w, s_{\min}) = \{I \in \mathcal{C}^*_{\mathcal{S}}(w, s_{\min}) \mid I \subseteq J\}$ be the subset of these patterns that are subsets of an item set J. Then we define the hypergraph connection strength of item $i \in J$ to the other items in J as

$$\xi^{(\text{pat})}_{\mathcal{S},w,s_{\min}}(i, J) = \sum_{I \in \mathcal{C}^*_{\mathcal{S},J}(w, s_{\min})} (|I| - r) \cdot s_{\mathcal{S},w}(I),$$

where $r \in \{0, 1\}$ is a parameter that determines whether the full pattern size (hyperedge size) should be considered ($r = 0$), or whether the item i itself should be disregarded ($r = 1$). The support of the item set I enters the definition because a larger support clearly means a stronger connection. Intuitively, $\xi^{(\text{pat})}_{\mathcal{S},w,s_{\min}}(i, J)$ sums the numbers of events underlying each of the patterns that connect i to the other items in J. Note that in this definition we assume (as is common practice) that $\xi^{(\text{pat})}_{\mathcal{S},w,s_{\min}}(i, J) = 0$ if $\mathcal{C}^*_{\mathcal{S},J}(w, s_{\min}) = \emptyset$.

This approach has the advantage that merely the filtered set of closed frequent patterns is needed. However, it has the disadvantage that subset patterns which, by chance, occur again outside of the instances of the full pattern may deteriorate the quality. An example of such an occurrence can be seen in Figure 1: the neurons a, b and e fire together between the second and third instance of the full set. However, this is not an incomplete instance of the full set, but rather a chance coincidence resulting from the background spikes. This can lead to a subset being preferred to the full pattern, even though the sum in the above definition gives higher weight to events that support multiple instances (as these are counted multiple times).

[1]While in a standard graph any edge connects exactly two vertices, in a hypergraph a single hyperedge can connect arbitrarily many vertices.

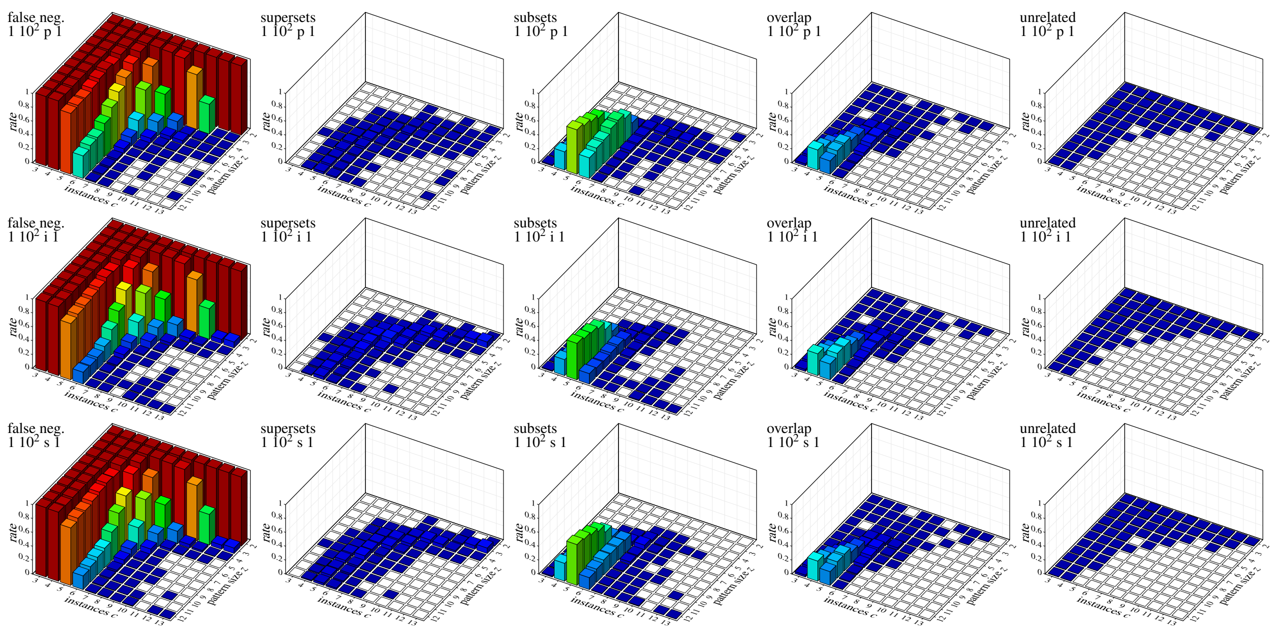

Figure 2: Experimental results with $\nu = 1$ (each item missing from one instance).

In order to improve on such cases, we refine our method by looking at the instances of each pattern:

Instance-based approach

Let $\mathcal{C}^*_{\mathcal{S}}(w, s_{\min})$ and $\mathcal{C}_{\mathcal{S},J}(w, s_{\min})$ be defined as above. Let $\mathcal{U}_{\mathcal{S},w}(I) \subseteq \mathcal{E}_{\mathcal{S},w}(I)$ be the independent set of instances of I that was identified by the CoCoNAD algorithm in order to compute the support $s_{\mathcal{S},w}(I)$. Furthermore, let $\mathcal{V}_{\mathcal{S},w,s_{\min}}(J) = \bigcup_{I \in \mathcal{C}^*_{\mathcal{S},J}(w,s_{\min})} \mathcal{U}_{\mathcal{S},w}(I)$. That is, $\mathcal{V}_{\mathcal{S},w,s_{\min}}(J)$ is the set of all instances underlying all patterns found in $\mathcal{S}$ that are subsets of J.

Our idea is that we only want to consider instances that are not “isolated”, but “overlap” some other instance (preferably of a different pattern). The reason is that isolated instances likely stem from chance coincidences, while instances that “overlap” other instances likely stem from the same (complete or incomplete) instance of the full pattern we try to identify. To implement this idea, we define

$$\begin{aligned}
&\mathcal{V}^*_{\mathcal{S},w,s_{\min}}(i, J) = \\
&\qquad \{\mathcal{R} \in \mathcal{V}_{\mathcal{S},w,s_{\min}}(J) \mid \exists t \colon \langle i, t\rangle \in \mathcal{R} \;\wedge \\
&\qquad\quad \exists \mathcal{T} \in \mathcal{V}_{\mathcal{S},w,s_{\min}}(J) \colon\; \mathcal{T} \neq \mathcal{R} \wedge o(\mathcal{T}, \mathcal{R})\},
\end{aligned}$$

where $o(\mathcal{R}, \mathcal{T})$ is an operator that tests whether the instances $\mathcal{R}$ and $\mathcal{T}$ overlap. In words: $\mathcal{V}^*_{\mathcal{S},w,s_{\min}}(i, J)$ is the set of instances of patterns that contain the item $i \in J$ and are subsets of the set J, which overlap at least one other instance.

For the operator o we tried two different variants:

$$\begin{aligned}
o_i(\mathcal{R}, \mathcal{T}) &= \begin{cases} 1 & \text{if } \mathcal{R} \cap \mathcal{T} \neq \emptyset, \\ 0 & \text{otherwise, and} \end{cases} \\
o_s(\mathcal{R}, \mathcal{T}) &= \sigma_w(\mathcal{R} \cup \mathcal{T}),
\end{aligned}$$

where σ_w is the synchrony operator from Section 2. That is, o_i checks whether the instances have a non-empty intersection, while o_s only checks whether the events underlying the instances are synchronous.

Based on these definitions, we finally define

$$\begin{aligned}
&\xi^{(\text{inst})}_{\mathcal{S},w,s_{min}}(i, J) = \\
&\qquad \left|\left\{\langle j, t\rangle \in \textstyle\bigcup_{\mathcal{R} \in \mathcal{V}^*_{\mathcal{S},w,s_{min}}(i,J)} \mathcal{R} \mid j \neq i \vee r = 0\right\}\right|,
\end{aligned}$$

where the parameter $r \in \{0, 1\}$ determines whether events of the item i should be considered ($r = 0$) or disregarded ($r = 1$). That is, the parameter r has the same function as the parameter r in the pattern-based approach (justifying the same name). Intuitively, $\xi^{(\text{inst})}_{\mathcal{S},w,s_{min}}(i, J)$ is the total number of events (possibly ignoring events of the item i) underlying instances that connect it to other items in J.

Compared to the pattern-based approach, the instance-based approach has the advantage that no events are counted multiple times and that chance coincidences are much less likely to deteriorate the detection quality. However, its disadvantage is that it is more costly to compute, because not just the patterns themselves, but the individual instances of all relevant patterns have to be processed.

6. Experiments

We implemented our detection method in Python, using an efficient C-based Python extension module that implements the CoCoNAD algorithm [2] as well as pattern spectrum estimation [3], while the reduction sequence was constructed directly in Python (see below for the sources). We generated event sequence data as independent Poisson processes with parameters chosen in reference to our application domain: 100 items (number of neurons that can be simultaneously recorded with current technology), 20Hz event rates (typical average firing rate observed in spike train recordings), 3s total time (typical recording times for spike trains range from a few seconds up to about an hour).

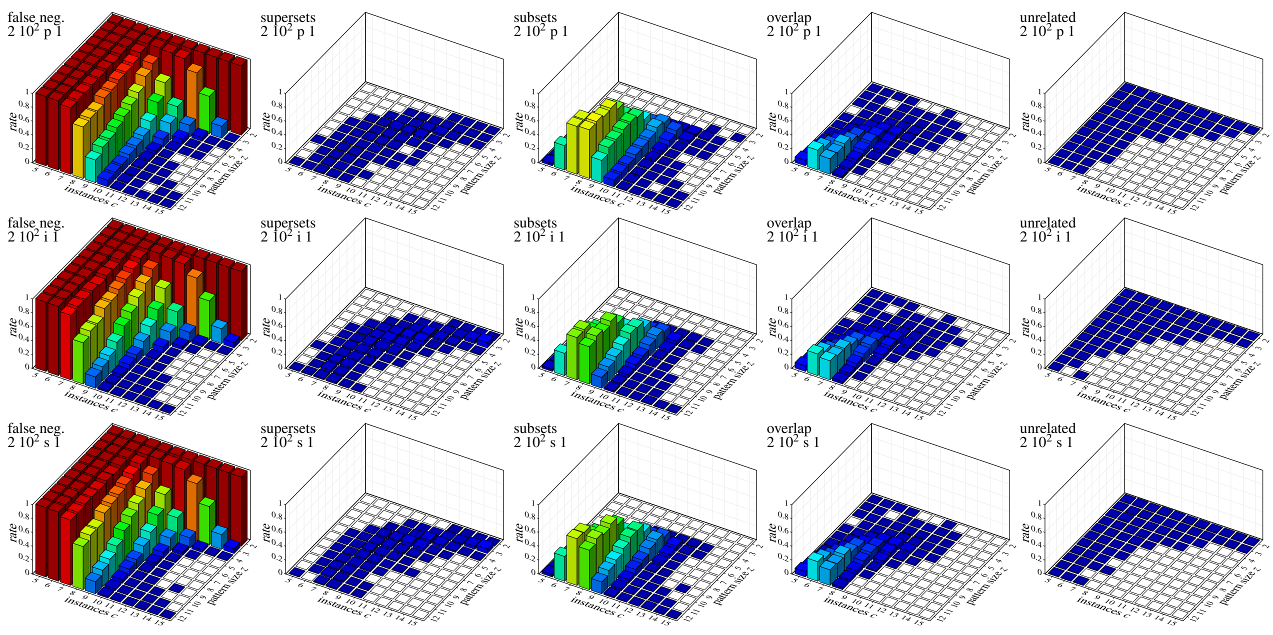

Figure 3: Experimental results with $\nu = 2$ (each item missing from two instances)

Into such independent data sets we injected a single parallel episode each, with sizes z ranging from 2 to 12 items and numbers c of occurrences (instances) ranging from 2 to 21. To simulate temporal imprecision, the events of each instance were jittered independently by drawing an offset from a uniform distribution on $[-1.5\text{ms}, +1.5\text{ms}]$ (which corresponds to typical bin lengths for time-binning of parallel neural spike trains, which are 1 to 7ms). To simulate selective participation, we deleted each item of a parallel episode from a number $\nu \in \{1, 2, 3, 4, 5\}$ of their instances. This created data sets with instances similar to those shown in Figure 1 (which corresponds to $z = 6$, $c = 6$ and $\nu = 1$, but has much fewer background spikes): a few instances may be complete, but most lack a varying number of items. For each signature $\langle z, c\rangle$ of a parallel episode and each value of ν we created 1000 such data sets.

Then we tried to detect the injected parallel episodes with the methods described in Section 5. For mining closed frequent patterns we used a window width of $w = 3$ms (matching the jitter of the temporal imprecision) as well as a minimum support $s_{\min} = 2$ and a minimum pattern size of 2 (to allow even for small patterns to be detected). Pattern spectrum filtering was executed with an estimated pattern spectrum, equivalent to 100 surrogate data sets. The resulting filtered closed patterns were subjected to all variants of the reduction sequence construction and result selection described in Section 5.

Some of the results we obtained are depicted in Figures 2 (for $\nu = 1$), 3 (for $\nu = 2$), and 4 (for $\nu = 4$). In each of these figures, the top row refers to the pattern-based approach (with $r = 1$), the second and third row to the instance-based approach (also with $r = 1$), where the second row uses the overlap operator o_i and the third row the overlap operator o_s. In each row, the first diagram shows the number of (strict) false negatives, that is, the fraction of runs (of 1000) in which something else than exactly the injected pattern was found.

In order to elucidate what happens in those runs in which the injected parallel episode was not (exactly) detected, the diagrams in columns 2 and 3 show the fraction of runs in which a superset or a subset, respectively, of the injected parallel episode was returned. Column 4 shows the fraction of runs with overlap patterns (the reported pattern contains some, but not all of the items of the injected parallel episode and at least one other item), column 5 the fraction of runs with patterns that are unrelated to the injected parallel episode. In all cases not recorded in columns 2 to 5, and not yielding exactly the injected parallel episode either, an empty pattern was returned by our procedure (not shown separately). Such a result (an empty pattern) is caused exclusively by the closed pattern mining not producing any patterns our reduction sequence construction can work on. That is, in these cases the synchronous activity in the data was not strong enough to distinguish it from chance patterns.

Among those cases in which a non-empty pattern is returned, that does not coincide with the injected parallel episode, subsets are most common (column 3), followed by overlap patterns. These cases mainly occur for lower numbers of instances, which is to be expected, as the smaller number of instances deteriorates the pattern structure. Once the number of instances is large enough, almost all runs produce exactly the injected parallel episode.

If we compare the different rows of each of the Figures 2–4, we see that the instance-based approach performs slightly better than the pattern-based approach, and the more so, the more events are missing. The two instance-based approaches (distinguished by the overlap operator: o_i or o_s)

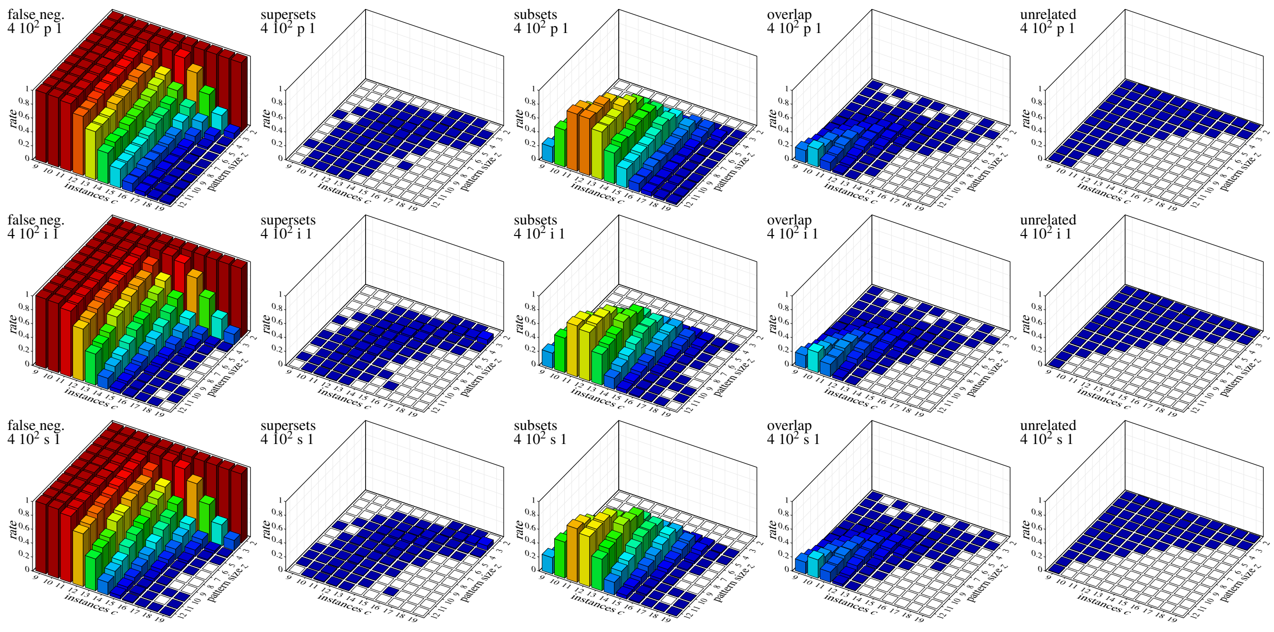

Figure 4: Experimental results with $\nu = 4$ (each item missing from four instances)

are essentially tied, possibly with a very slight advantage for the overlap operator o_i.

If we compare the diagrams across the three Figures 2–4, we may also conjecture that each additional instance from which items are missing, requires about two additional instances to compensate the reduction in detection quality (note the different scales on the instance axis!). This is also plausible, since each item missing from one additional instance effectively removes an instance (as it removes as many events as an instance contains) and since the removals are distributed over multiple instances, additional compensation is needed. Furthermore, we see that we achieve reliable detection even if items are missing from up to about one quarter of the instances of a pattern.

In summary, our methods appear to work very well, given how little structure is present in the data, especially for larger values of ν. If our method detects something at all (which is the case as soon as there is structure in the data that cannot be explained as a chance event), it is almost always related to the actual parallel episode, and most often the exact parallel episode, provided the number of instances is large enough.

7. Conclusion

In this paper, we presented a method to detect parallel episodes in point processes in the presence of both temporal imprecision and selective participation. This method is based on finding (partial) patterns with the CoCoNAD methodology and then trying to combine these patterns by finding a densely connected sub-hypergraph in the hypergraph that is induced by these patterns on the set of items. Our extensive experiments demonstrate that the method works very well, although about two additional instances are needed to compensate each item missing from one additional instance. This is not surprising, though, considering the loss of information that results from the missing items. Rather it is surprising that only two additional coincidences are enough to allow fairly reliable detection.

Software

An implementation of the CoCoNAD algorithm in Python and in C (as a stand-alone program and as a Python extension library) can be found at
 http://www.borgelt.net/coconad.html and
 http://www.borgelt.net/pycoco.html.
A Java graphical user interface is available at
 http://www.borgelt.net/cocogui.html.
The scripts with which we executed our experiments as well as the complete result diagrams (all parameter combinations) have been made available at
 http://www.borgelt.net/hypernad.html.

Acknowledgments

The work presented in this paper was partially supported by the Spanish Ministry for Economy and Competitiveness (MINECO Grant TIN2012-31372) and by the Government of the Principality of Asturias (Programa Asturias Grant CT14-05-2-06).

References

[1] C. Borgelt. Frequent Item Set Mining. *Wiley Interdisciplinary Reviews (WIREs): Data Mining and Knowledge Discovery* 2:437–456 (doi:10.1002/widm.1074). J. Wiley & Sons, Chichester, United Kingdom 2012

[2] C. Borgelt and D. Picado-Muiño. Finding Frequent Synchronous Events in Parallel Point

Processes. *Proc. 12th Int. Symposium on Intelligent Data Analysis (IDA 2013, London, UK)*, 116–126. Springer-Verlag, Berlin/Heidelberg, Germany 2013

[3] C. Borgelt and D. Picado-Muiño. Simple Pattern Spectrum Estimation for Fast Pattern Filtering with CoCoNAD. *Proc. 13th Int. Symposium on Intelligent Data Analysis (IDA 2014, Leuven, Belgium)*, 37–48. Springer-Verlag, Berlin/Heidelberg, Germany 2014

[4] M. Fiedler and C. Borgelt. Subgraph Support in a Single Graph. *Proc. IEEE Int. Workshop on Mining Graphs and Complex Data*, 399–404. IEEE Press, Piscataway, NJ, USA 2007

[5] J. Høastad. Clique is Hard to Approximate within n^{1-e}. *Acta Mathematica* 182:105–142. Mittag–Leffler Institute, Stockholm, Sweden 1999

[6] D.O. Hebb. *The Organization of Behavior.* J. Wiley & Sons, New York, NY, USA 1949

[7] R.M. Karp. Reducibility among Combinatorial Problems. In: R.E. Miller and J.W. Thatcher (eds.) *Complexity of Computer Computations*, 85–103. Plenum Press, New York, NY, USA 1972

[8] S. Laxman, P.S. Sastry, and K. Unnikrishnan. Discovering Frequent Episodes and Learning Hidden Markov Models: A Formal Connection. *IEEE Trans. on Knowledge and Data Engineering* 17(11):1505–1517. IEEE Press, Piscataway, NJ, USA 2005

[9] S. Louis, C. Borgelt, and S. Grün. Generation and Selection of Surrogate Methods for Correlation Analysis. In: S. Grün and S. Rotter (eds.) *Analysis of Parallel Spike Trains*, 359–382. Springer-Verlag, Berlin, Germany 2010

[10] H. Mannila, H. Toivonen, and A. Verkamo. Discovery of Frequent Episodes in Event Sequences. *Data Mining and Knowledge Discovery* 1(3):259–289. Springer, New York, NY, USA 1997

[11] D. Picado-Muiño, C. Borgelt, D. Berger, G.L. Gerstein, and S. Grün. Finding Neural Assemblies with Frequent Item Set Mining. *Frontiers in Neuroinformatics* 7:article 9 (doi:10.3389/fninf.2013.00009). Frontiers Media, Lausanne, Switzerland 2013

[12] D. Picado-Muiño and C. Borgelt. Frequent Itemset Mining for Sequential Data: Synchrony in Neuronal Spike Trains. *Intelligent Data Analysis* 18(6):997-1012. IOS Press, Amsterdam, Netherlands 2014

[13] N. Tatti. Significance of Episodes Based on Minimal Windows. *Proc. 9th IEEE Int. Conf. on Data Mining (ICDM'09, Miami, FL, USA)*, 513–522. IEEE Press, Piscataway, NJ, USA 2009

[14] E. Torre, D. Picado-Muiño, M. Denker, C. Borgelt, and S. Grün. Statistical Evaluation of Synchronous Spike Patterns Extracted by Frequent Item Set Mining. *Frontiers in Computational Neuroscience*, 7:article 132 (doi:10.3389/fninf.2013.00132). Frontiers Media, Lausanne, Switzerland 2013

[15] C. Tsourakakis, F. Bonchi, A. Gionis, F. Gullo, and M. Tsiarli. Denser than the Densest Subgraph: Extracting Optimal Quasi-Cliques with Quality Guarantees. *Proc. 19th ACM SIGMOD Int. Conf. on Knowledge Discovery and Data Mining (KDD 2013, Chicago, IL)*, 104–112. ACM Press, New York, NY, USA 2013

[16] N. Vanetik, E. Gudes, and S.E. Shimony. Computing Frequent Graph Patterns from Semistructured Data. *Proc. IEEE Int. Conf. on Data Mining*, 458–465. IEEE Press, Piscataway, NJ, USA 2002

[17] M.J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. New Algorithms for Fast Discovery of Association Rules. *Proc. 3rd Int. Conf. on Knowledge Discovery and Data Mining (KDD 1997, Newport Beach, CA)*, 283–296. AAAI Press, Menlo Park, CA, USA 1997