Mining Significant Frequent Patterns in Parallel Episodes with a Graded Notion of Synchrony and Selective Participation

Salatiel Ezennaya-Gomez[1,2] and Christian Borgelt[1]
[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, Magdeburg, Germany*

Keywords: Graded Synchrony, Synchronous Events, Temporal Imprecision, Selective Participation, Frequent Pattern Mining.

Abstract: We consider the task of finding frequent parallel episodes in parallel point processes (or event sequences), allowing for imprecise synchrony of the events constituting occurrences (temporal imprecision) as well as incomplete occurrences (selective participation). The temporal imprecision problem is tackled by frequent pattern mining using a graded notion of synchrony that captures both the number of instances of a pattern as well as the precision of synchrony of its events. To cope with selective participation, a reduction sequence of items (or event types) is formed based on found frequent patterns and guided by pattern overlap. We evaluate the performance of this method on a large number of data sets with injected parallel episodes. We demonstrate that, in contrast to binary synchrony where it pays to consider the pattern instances, graded synchrony performs better with a pattern-based scheme than with an instance-based one, thus simplifying the procedure.

1 INTRODUCTION

We present elaborate methodology to identify meaningful frequent synchronous patterns in event sequences (see e.g. (Mannila et al., 1997)), using the principles of frequent item set mining (FIM) (see e.g. (Borgelt, 2012)). As is well known, the objective of FIM is to find all item sets that are frequent in a transaction database. FIM uses the support (number of occurrences in the transactions) to define an item set as frequent, namely if its support reaches or exceeds a (user-specified) minimum support threshold. In standard FIM the support of an item set is a simple count of transactions, whereas in our case the event sequence data is continuous in nature, since it resides in the time domain, and thus no (natural) transactions exist. This continuous form causes several problems, especially w.r.t. the definition of a support measure.

Frequent pattern mining in continuous time faces two main problems: *temporal imprecision* and *selective participation*. The former is related to the synchrony of events which is affected by temporal jitter, due to which the events are usually not perfectly aligned. In frequent pattern mining we tackle temporal imprecision by defining that items (or events) co-occur if they occur in a (user-specified) limited time span from each other. In earlier work the size of a maximum independent set (MIS) of such synchronous groups of items, which can be computed efficiently with a greedy algorithm (see (Borgelt and Picado-Muino, 2013) and (Picado-Muino and Borgelt, 2014)), is used as a support measure. This method is referred to as *binary synchrony* in (Ezennaya-Gómez and Borgelt, 2015), because it allows only for two values: either a group of events is synchronous, or it is not.

Unfortunately, a greedy algorithm no longer guarantees an optimal solution when a *graded notion of synchrony* is introduced, while a backtracking approach (as it would be used for a general maximum independent set problem, which is NP-complete) takes exponential time in the worst case. As a consequence, an adaptation is necessary, which takes the form of an approximation procedure to compute the support, but maintaining the crucial property of support being *anti-monotone*. To this end, (Ezennaya-Gómez and Borgelt, 2015) defined a graded synchrony approach where the support computation takes the precision of synchrony into account. That is, a pattern that has fewer occurrences, but in each of these the items occur very closely together in time, is rated better than an item set, which has more instances, but in each

Ezennaya-Gomez, S. and Borgelt, C..
Mining Significant Frequent Patterns in Parallel Episodes with a Graded Notion of Synchrony and Selective Participation.
In *Proceedings of the 7th International Joint Conference on Computational Intelligence (IJCCI 2015) - Volume 3: NCTA*, pages 39-48
ISBN: 978-989-758-157-1

of these the synchrony of the events is rather loose (Ezennaya-Gómez and Borgelt, 2015).

The second problem, that is, *selective participation*, is related to the lack of occurrence of some items, which produces incomplete pattern instances. As a consequence, only subsets of the actual pattern are present in the instances underlying a pattern. This can be caused by imperfections of the measuring technology or by properties of the underlying process. (Borgelt et al., 2015) presented an approach to solve this problem in the binary synchrony setting we mentioned above. The goal of this paper is to transfer this approach from the binary synchrony setting to a graded synchrony setting, and to investigate whether the same conclusions can be drawn about the different variants considered in (Borgelt et al., 2015).

The application area that motivates our work is the analysis of *parallel spike trains* in neurobiology. Such spike trains are sequences of points in time, one per neuron, that represent the times at which an electrical impulse (*action potential* or *spike*) is emitted. The objective is to identify *cell assemblies* or groups of neurons that tend to exhibit synchronous spiking. Such cell assemblies were proposed in (Hebb, 1949) as a model for encoding and processing information in biological neural networks. Since synchronous spike input to receiving neurons is known to be more effective in generating output spikes (Abeles, 1982; König et al., 1996), such cell assemblies are a plausible hypothesis for neural information processing.

The objective of this paper is to adapt the *selective participation approach* in an algorithm called CoCoNAD (for **Co**ntinuous-time **Clo**sed **N**euron **A**ssembly **D**etection, a name inspired by the mentioned application domain) to detect significant synchronous patterns with a *graded notion of synchrony*.

As a first step to identify 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 are expected to be present and thus require proper treatment. Once frequent patterns are detected, additional filtering is necessary to remove those frequent patterns that are produced by chance events and thus are not relevant. Then, detection of selective participation is applied. Due to the graded nature of the synchrony definition we adopt, the selective participation approach needs to be modified from the approach presented in (Borgelt et al., 2015).

The remainder of this paper is structured as follows: Section 2 covers basic terminology and notation and introduces our graded notion of synchrony. In Sections 3 and 4 we show how a support based on graded synchrony is approximated and frequent synchronous patterns are mined and filtered using pattern spectrum filtering. In Section 5 we present our methodology to identify frequent parallel episodes with selective participation with a graded notion of synchrony. Section 6 reports experimental results on data sets with injected parallel episodes. Finally, in Section 6 we draw conclusions from our discussion.

2 EVENT SEQUENCES & SYNCHRONY

We adopt notation and terminology from (Mannila et al., 1997) and (Ezennaya-Gómez and Borgelt, 2015). The data are sequences of events $\mathcal{S} = \{\langle i_1,t_1\rangle,\ldots,\langle i_m,t_m\rangle\}$, $m \in \mathbb{N}$, 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, $k \in \{1,\ldots,m\}$. 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. Such data may as well be represented as *parallel point processes* $\mathcal{P} = \{\langle i_1,\{t_1^{(1)},\ldots,t_{m_1}^{(1)}\}\rangle,\ldots,\langle i_n,\{t_1^{(n)},\ldots,t_{m_n}^{(n)}\}\rangle\}$ 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 (Ezennaya-Gómez and Borgelt, 2015).

A *synchronous pattern* (in $\mathcal{S}$) is defined as a set of items $I \subseteq B$ that occur several times (approximately) synchronously in $\mathcal{S}$. Formally, an *instance* (or *occurrence*) of such a synchronous pattern (or a set of *synchronous events for I*) in an event sequence $\mathcal{S}$ with respect to a user-specified time span $w \in \mathbb{R}^+$ is 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. Let ϕ the pattern operator that yields the pattern underlying an instance, $\phi(\mathcal{R}) = \{i \mid \langle i,t\rangle \in \mathcal{R}\}$. Hence the set of all instances of a pattern $I \subseteq B$, $I \neq \emptyset$, in an event sequence $\mathcal{S}$ is

$$\mathcal{E}_{\mathcal{S},w}(I) = \big\{\mathcal{R} \subseteq \mathcal{S} \mid \phi(\mathcal{R}) = I \wedge |\mathcal{R}| = |I| \wedge \sigma_w(\mathcal{R}) > 0\big\},$$

where σ_w is the synchrony operator which measures the degree of synchrony of the events in $\mathcal{R}$.

The synchrony operator should conincide with binary synchrony for limiting cases as follows: if all events in $\mathcal{R}$ coincide (i.e. have exactly the same oc-

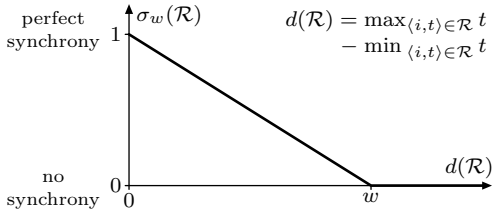

Figure 1: Degree of synchrony as a function of the distance between the latest and the earliest event.

currence time, perfect synchrony), the degree of synchrony should be 1, while it should be 0 if the events are spread out farther than the window width w (no synchrony). However, if the (time) distance between the earliest and the latest event in $\mathcal{R}$ is between 0 and w, we want a degree of synchrony between 0 and 1.

Such a synchrony operator was described in (Picado-Muino et al., 2012) based on the notion of an influence map, which is placed at each event and describes the vicinity around an event in which synchrony with other events is defined. Such an influence map for an event occurring at time t is defined as a function

$$f_t(x) = \begin{cases} \frac{1}{w} & \text{if } x \in [t - \frac{w}{2}, t + \frac{w}{2}], \\ 0 & \text{otherwise.} \end{cases}$$

Based on influence maps, events are synchronous iff their influence maps overlap. The area of the overlap measures the degree of synchrony (Figure 1):

$$\sigma_w(\mathcal{R}) = \int_0^\infty \min_{\langle i,t\rangle\in R} f_t(x,w)\, \mathrm{d}x.$$

Alternatively, we may use the equivalent definition

$$\sigma_w(\mathcal{R}) = \max\left\{0, 1 - \tfrac{1}{w}\left(\max_{\langle i,t\rangle\in\mathcal{R}} t - \min_{\langle i,t\rangle\in\mathcal{R}} t\right)\right\}.$$

The synchrony operator underlies the definition of a support operator $s_{\mathcal{S},w}(I)$ that is used to mine synchronous patterns. The support should capture (also) the number of occurrences of a pattern in a given event sequence $\mathcal{S}$. In addition, in order to be efficient, frequent pattern mining requires support to be *anti-monotone*: $\forall I \subseteq J \subseteq B : s_{\mathcal{S},w}(I) \geq s_{\mathcal{S},w}(J)$. Or in words: *if an item is added to an item set, its support cannot increase*. This implies the so-called *apriori property*: $\forall I, J \subseteq B : (J \supseteq I \wedge s_{\mathcal{S},w}(I) < s_{\min}) \Rightarrow s_{\mathcal{S},w}(J) < s_{\min}$. Or in words: *no superset of an infrequent pattern can be frequent*. The apriori property allows to prune the search for frequent patterns effectively (Borgelt, 2012).

Given a support measure and a (user-specified) minimum support $s_{\min}$, the task of frequent synchronous pattern mining is defined as the task to identify all item sets $I \subseteq B$ with $s_{\mathcal{S},w}(I) \geq s_{\min}$ ((Borgelt, 2012)).

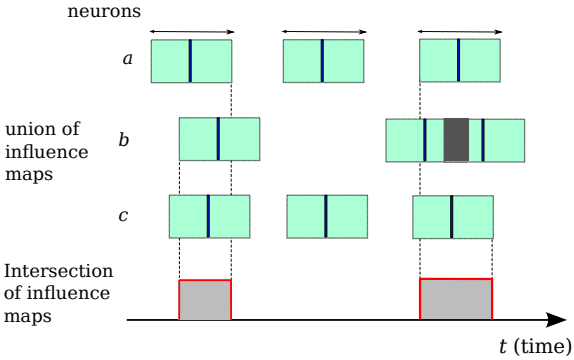

Figure 2: Support computation for three items a, b, c. Each event has its influence map (represented as a rectangle). If two influence maps overlap, the resulting influence map is the maximum (union) of these influence maps. The intersection of influence maps is the minimum which defines the synchrony operator. In the diagram, item b has two events the influence regions of which overlap. The support results from the integral over the intersections.

3 SUPPORT COMPUTATION FOR GRADED SYNCHRONY

For binary synchrony the support is computed efficiently using a greedy algorithm. However, following (Ezennaya-Gómez and Borgelt, 2015) and as we mentioned in the introduction, for graded synchrony such an approach does not guarantee an optimal result and thus needs to be replaced by an approximation.

As such an approximation, defined in (Ezennaya-Gómez and Borgelt, 2015), the integral over the maximum (union) of the minimum (intersection) of influence regions is chosen: the minimum represents the synchrony operator, the maximum takes care of a possible overlap between instances of synchronous event groups, and the integral finally aggregates over different instances. Formally:

$$s_{\mathcal{S},w}(I) = \int_{-\infty}^{\infty} \max_{\mathcal{R}\in\mathcal{T}_{\mathcal{S},w}(I)} \left(\min_{\langle i,t\rangle\in\mathcal{R}} f_t(x)\right) \mathrm{d}x.$$

The advantages of this support measure are mainly two: in the first place, the support measure defined in this way is anti-monotone due to the minimum over $i \in I$. Secondly, it allows to compute the support by a simple intersection of interval lists, since all occurring functions only take two values, namely 0 and $\frac{1}{w}$, and therefore it suffices to record where they are greater than 0. Thus, the list of intervals for each item $i \in B$ in which $\max_{\langle j,t\rangle\in\mathcal{S};j=i} f_t(x) > 0$ is computed. These intervals can then be intersected to account for the minimum. Summing the interval lengths and dividing by w we obtain the area under the functions (see the example described in Figure 2).

Note that this computation scheme is very similar to the *Eclat algorithm* (Zaki et al., 1997) (which works by intersecting transaction identifier lists), transferred to a continuous domain (and thus to effectively infinitely many transactions). As a consequence, Eclat's item set enumeration scheme, which is based on a simple *divide-and-conquer* approach, can be applied with only few adaptations (concerning mainly the support computation) to obtain an efficient algorithm for mining frequent synchronous patterns (see e.g. (Borgelt, 2012)).

4 PATTERN SPECTRUM FILTERING

The large number of patterns in the output of synchronous pattern mining method is a problem and thus further reduction is necessary. This is done by identifying statistically significant patterns. Previous work showed that statistical tests on individual patterns are not suitable (Picado-Muino et al., 2013; Torre et al., 2013). The main problems are the lack of proper test statistics as well as *multiple testing*, that is, the huge number of patterns makes it very difficult to control the family-wise error rate, even with control methods like *Bonferroni correction*, the *Benjamini-Hochberg procedure* or the *false discovery rate* etc. (Dudoit and van der Laan, 2008).

To overcome this problem, we rely here on the approach suggested in (Picado-Muino et al., 2013) and refined in (Torre et al., 2013), namely *Pattern Spectrum Filtering* (PSF). 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. From this insight it was derived in (Picado-Muino et al., 2013) that patterns should rather be judged based on their *signature* $\langle z,s \rangle$, where $z = |I|$ is the size of a pattern I and s its support. A pattern is not significant if a counterpart which has the same or larger pattern size z and same or higher support s can be explained as a chance event under the null hypothesis of independent events.

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

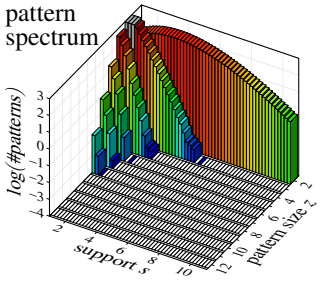

Figure 3: Pattern spectrum generated from 10^4 surrogate data sets.

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 several surrogate data methods in the context of neural spike train analysis can be found in (Louis et al., 2010).

Summarizing, the objective of *PSF* is to pool the resulting patterns from the output of synchronous pattern mining with the same or larger signature $\langle z,s \rangle$ filtering by the surrogate data sets generated as an implicit representation of the null hypothesis. An example of such a pattern spectrum, for data as it will be used in Section 6, is shown in Figure 3. It captures what pattern signatures occurred in a large number of surrogate data sets. Note that since we are working in continuous time domain the support values are (non-negative) real numbers.

5 SELECTIVE PARTICIPATION METHOD

To achieve the purpose of this paper we draw in the same idea described in (Borgelt et al., 2015) to identify parallel episodes in the presence of selective participation. The approach 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.* An example of this situation is illustrated in Figure 4, 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 (leftmost 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 sub-

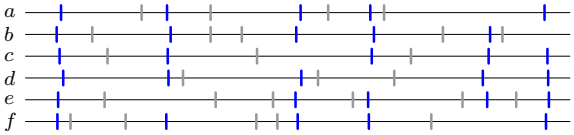

Figure 4: Parallel episodes (indicating neuron assembly activity) with selective participation (blue) as well as background spikes (gray).

sets 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.

The method views the set of patterns that were 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, (Borgelt et al., 2015).

The method draws on the approach proposed in (Tsourakakis et al., 2013) for detecting dense sub-hypergraphs. 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, (Borgelt et al., 2015).

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 (Tsourakakis et al., 2013): 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.

Formally, a reduction sequence of item sets is constructed, 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). Then, we assign a quality measure to each element of this reduction sequence:

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

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 $\xi_{\mathcal{S},w,s_{\min}}(i,J_k)$, two different approaches were defined in (Borgelt et al., 2015): a *pattern-based approach* which works with patterns and support and *instance-based approach* which tries to remove instances to focus the evaluation on instances that likely resulted from the actual assembly activity. These methods are described for binary synchrony. To apply them to our graded synchrony, a re-definition of the latter was necessary and it is described further down.

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 with a graded notion of synchrony (if executed with window width w and minimum support $s_{\min}$ on $\mathcal{S}$), for which no counterpart (no signature with the same size and greater or equal support constitutes a counterpart) 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.

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

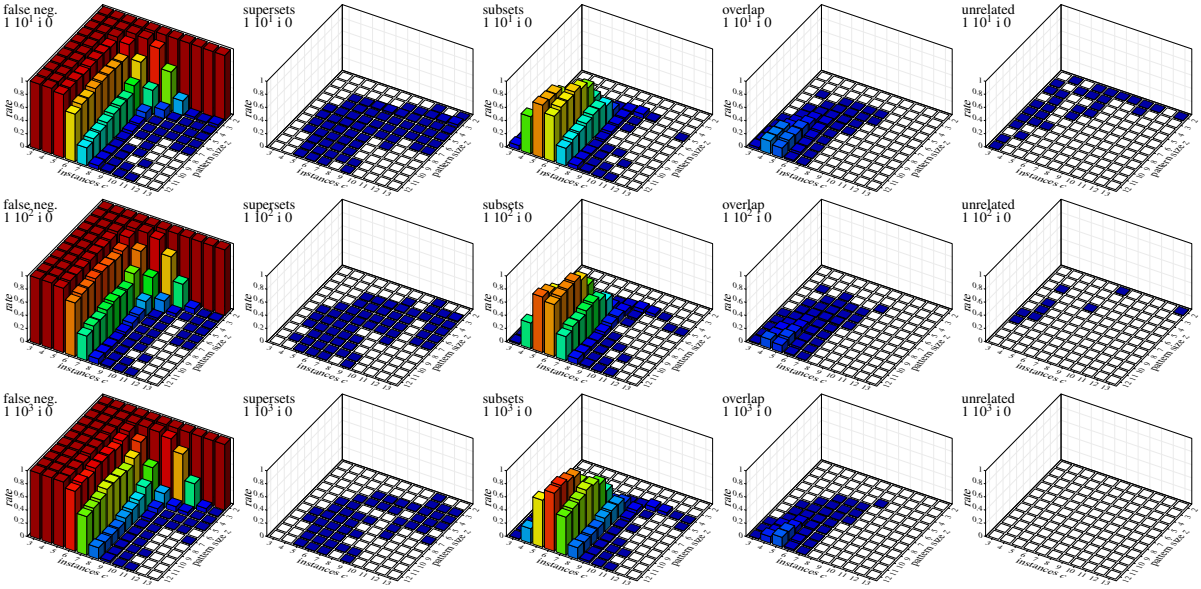

Figure 5: Experimental results with $\nu = 1$ (each item missing from one instance). Patterns filtered by 10, 100, and 1000 surrogate data sets.

Intuitively, $\xi^{(\text{pat})}_{\mathcal{S},w,s_{\min}}(i,J)$ sums the grades of synchrony 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 detection quality. An example of such an occurrence can be seen in Figure 4: 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). Removing instances is necessary to improve the detection quality. To obtain concrete instances of the functions $\xi_{\mathcal{S},w,s_{\min}}(i,J_k)$ we define:

Instance-based Approach

We start from the same idea in (Borgelt et al., 2015) 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. Unlike (Borgelt et al., 2015) where the instance-based approach merely counts spikes without considering the precision of synchrony, in our approach we recompute the support from the reduced set of instances, which is simple as the instances are known.

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 set of all 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.

To implement our idea of keeping overlapping instances of different patterns, we define

$$\mathcal{V}^*_{\mathcal{S},w,s_{\min}}(i,J) = \{\mathcal{R} \in \mathcal{V}_{\mathcal{S},w,s_{\min}}(J) \mid \wedge \quad \exists \mathcal{T} \in \mathcal{V}_{\mathcal{S},w,s_{\min}}(J) : \phi(\mathcal{T}) \neq \phi(\mathcal{R}) \wedge o_i(\mathcal{T},\mathcal{R}) = 1\},$$

where ϕ is the pattern operator defined in Section 2 and $o_i(\mathcal{R},\mathcal{T})$ 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 of a different pattern. The operator o_i checks whether the instances have a non-empty intersection. The instance-based approach has the advantage 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, but the individual instances of all relevant patterns have to be processed.

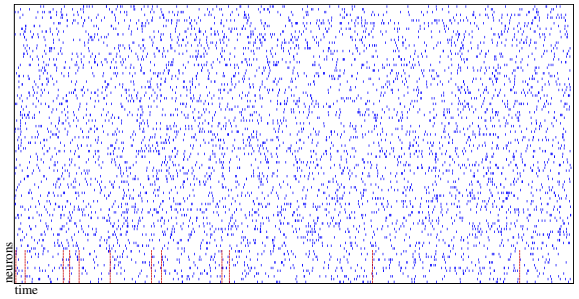

Figure 6: Example of generated data sets that imitate parallel neural spike trains. Each row of blue dots represents the spike train of each neuron. In this example the injected patterns are drawn in red.

The selective participation adaptation method proposes to study only instances that overlap with other instances of different patterns, re-compute the support for these patterns and finally apply the pattern-based approach.

6 EXPERIMENTS

We implemented our frequent synchronous pattern mining method in Python, using an efficient C-based Python extension module that implements the pattern mining and surrogate generation.[2] 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).

Into such independent data sets we injected a single synchronous pattern each, with sizes z ranging from 2 to 12 items and numbers c of occurrences (instances) ranging from 2 to 12. To simulate imprecise synchrony, the events of each pattern instance were jittered independently by drawing an offset from a uniform distribution on $[-1\text{ms}, +1\text{ms}]$ (which corresponds to typical bin lengths for time-binning of parallel neural spike trains, which are 1 to 7ms). An example of such a data set is depicted in Figure 6.

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 4 (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 synchronous patterns with the methods described in Sections 4 and 5. For mining closed frequent patterns we used different values of the window width, $w \in \{2,3,4,5\}$ (matching the jitter of the temporal imprecision) using a minimum support $s_{\min} = 1.0$ and a minimum pattern size $z_{\min} = 2$. Based on results presented in (Ezennaya-Gómez and Borgelt, 2015), the window value used is $\frac{3}{2}j$. Then filtering patterns mined with different pattern spectrum derived from 10, 100 and 1000 data sets with independent spike trains. The method described in Section 5 is applied to the resulting patterns.

Some of the results we obtained are depicted in Figures 5, 7, and 8. In each row of the figures, 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. On top of each diagram the different approaches are shown with its parameters. The first value corresponds to the number of missing instances followed by the type of filter, and the type of reduction sequence approach applied and value of r parameter.

Figure 5 shows the different results obtained by filtering closed frequent patterns with the different patterns spectra (10, 100 and 1000). Note that graded synchrony has less problems with unrelated patterns. We observe that filtering by surrogates derived from 100 and 1000 performs better w.r.t. unrelated patterns. That is, filtering with 100 and 1000 surrogate data sets detect every injected pattern if only something is detected at all. Figure 7 shows a comparison between

[2] The implementation of the CoCoNAD algorithm is developed in Python (Rossum, 1993) and the core of the algorithm is developed in C (Kernighan and Ritchie, 1978). This implementation 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) will be available at
http://www.borgelt.net/hypernad.html.

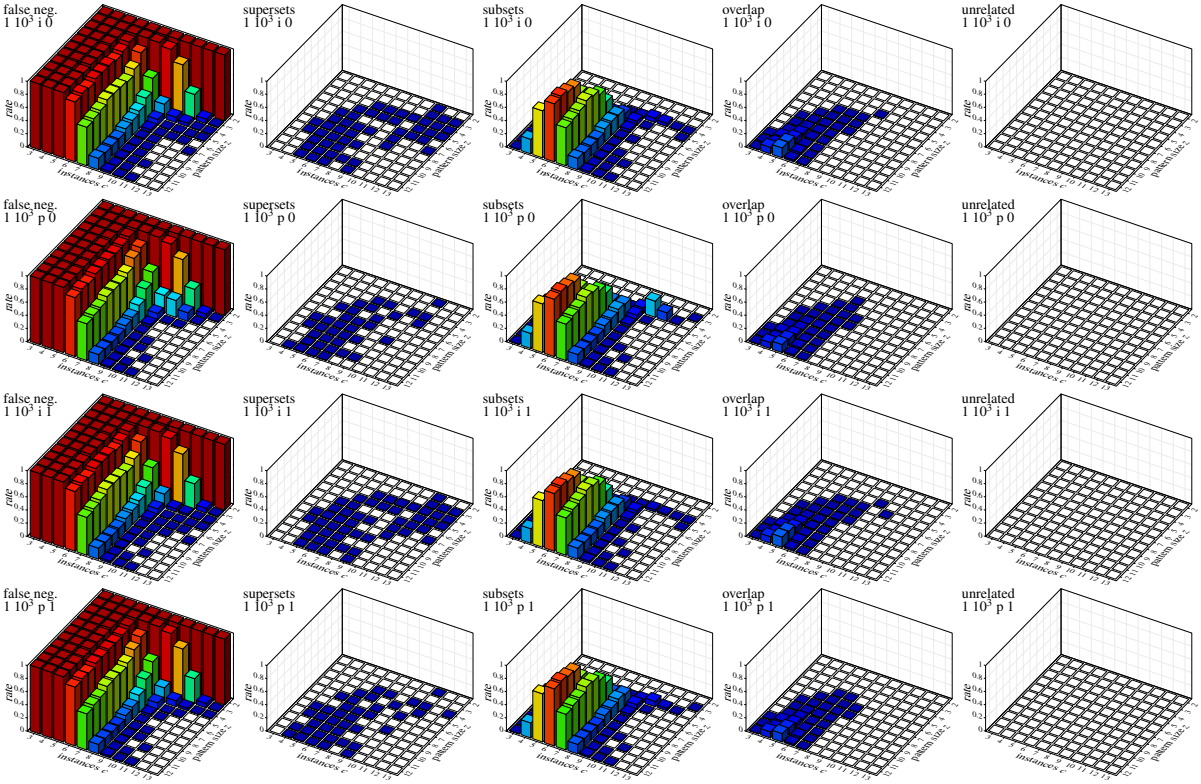

Figure 7: Experimental results with instance-based approach and pattern-based approach for graded synchrony (each item missing from one instance).

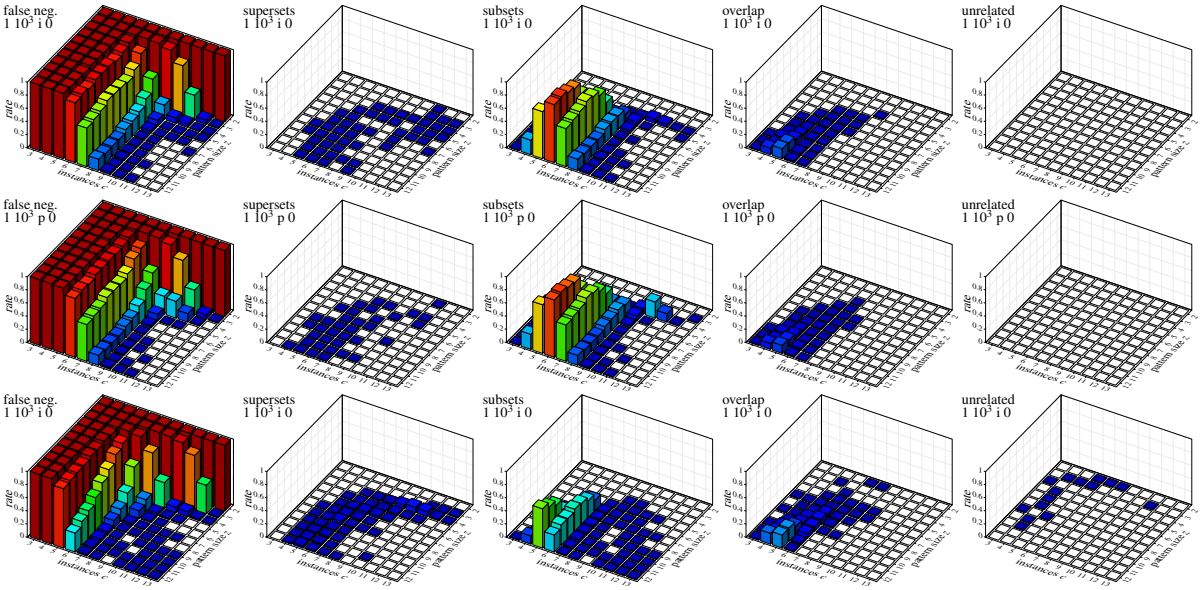

Figure 8: Comparison between the instance/pattern-based approach for graded synchrony and the instance-based approach for binary synchrony.

the instance-based approach and the pattern-based approach using graded synchrony. The firsts two rows of diagrams show results for $r = 0$ and the second two rows correspond to $r = 1$. The pattern-based approach is slightly better than the instance-based approach in terms of false negatives (exact pattern detection). Concretely, for $r = 1$, the pattern-based approach has better ratios in supersets and overlaps by

paying a price of a slightly worse ratio for subsets. Taken together, the price of having more subsets is preferred, because subsets contain only items actually in the assembly, while superset and overlap patterns also contain unrelated items.

The first and second row of figure 8 correspond to the instance and the pattern-based approach for graded synchrony. The third corresponds to the instance-based approach for binary synchrony. Comparing the diagrams for unrelated patterns, our graded method detects all injected patterns (first and second rows), while the binary method also produces unrelated pattern. In (Borgelt et al., 2015) it is demonstrated that the instance-based approach yields slightly better results than the pattern approach. However, this approach does not consider the precision of synchrony. Surprisingly, using only the pattern-based approach with a graded notion of synchrony yields a better ratio for overlap and superset patterns.

7 CONCLUSIONS

In this paper we presented a method to detect frequent synchronous patterns in event sequences using a graded notion of synchrony for mining patterns in the presence of imprecise synchrony of events constituting occurrences and selective participation (incomplete occurrences). Our method adapts methods presented in the literature to tackle selective participation using binary synchrony, especially the instance-based approach which looks at instances of patterns to improve the detection by removing instances that are likely chance events, checking the precision of synchrony of these instances. We demonstrate in our experiments that using a graded notion of synchrony for support computation helps to simplify the detection of selective participation, because a pattern-based approach yields better results or at least equally good results as an instance-based approach. This is a considerable advantage, since identifying the individual pattern instances is costly and thus it is desirable to avoid it.

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 Principality of Asturias, through the 2013-2017 Science Technology and Innovation Plan (Programa Asturias, CT1405206), and the European Union, through FEDER funds.

REFERENCES

Abeles, M. (1982). Role of the cortical neuron: Integrator or coincidence detector? *Israel Journal of Medical Sciences*, 18(1):83–92.

Borgelt, C. (2012). Frequent item set mining. In *Wiley Interdisciplinary Reviews (WIREs): Data Mining and Knowledge Discovery*, pages 437–456. (J. Wiley & Sons, Chichester, United Kingdom, 2.

Borgelt, C., Braune, C., and Loewe, K. (2015). Mining frequent parallel episodes with selective participation. In *Proc. 16th World Congress of the International Fuzzy Systems Association (IFSA) and 9th Conference of the European Society for Fuzzy Logic and Technology (EUSFLAT), IFSA-EUSFLAT2015*, Gijon, Spain. Atlantis Press.

Borgelt, C. and Picado-Muino, D. (2013). Finding frequent synchronous events in parallel point processes. In *Proc. 12th Int. Symposium on Intelligent Data Analysis (IDA 2013, London, UK)*, pages 116–126, Berlin/Heidelberg, Germany. Springer-Verlag.

Dudoit, S. and van der Laan, M. J. (2008). *Multiple Testing Procedures with Application to Genomics*. Springer, New York, USA.

Ezennaya-Gómez, S. and Borgelt, C. (2015). Mining frequent synchronous patterns with a graded notion of synchrony. In *Proc. 16th World Congress of the International Fuzzy Systems Association (IFSA) and 9th Conference of the European Society for Fuzzy Logic and Technology (EUSFLAT), IFSA-EUSFLAT2015*, pages 1338–1345, Gijon, Spain. Atlantis Press, ISBN (on-line): 978-94-62520-77-6.

Hebb, D. O. (1949). *The Organization of Behavior*. J. Wiley & Sons, New York, NY, USA.

Kernighan, W. and Ritchie, D. (1978). *The C Programming Language*. Prentice Hall.

König, P., Engel, A. K., and Singer, W. (1996). Integrator or coincidence detector? the role of the cortical neuron revisited. *Trends in Neurosciences*, 19(4):130–137.

Louis, S., Borgelt, C., and Grün, S. (2010). Generation and selection of surrogate methods for correlation analysis. In Grün, S. and Rotter, S., editors, *Analysis of Parallel Spike Trains*, pages 359–382. Springer-Verlag, Berlin, Germany.

Mannila, H., Toivonen, H., and Verkamo, A. (1997). Discovery of frequent episodes in event sequences. In *Data Mining and Knowledge Discovery*, pages 259–289. Springer, New York, NY, USA, 1(3).

Picado-Muino, D. and Borgelt, C. (2014). Frequent itemset mining for sequential data: Synchrony in neuronal spike trains. *Intelligent Data Analysis*, 18(6):997–1012.

Picado-Muino, D., Borgelt, C., Berger, D., Gerstein, G. L., and Grün, S. (2013). Finding neural assemblies with frequent item set mining. *Frontiers in Neuroinformatics*, 7.

Picado-Muino, D., Castro-León, I., and Borgelt, C. (2012). Fuzzy frequent pattern mining in spike trains. In *Proc. 11th Int. Symposium on Intelligent Data Analysis (IDA 2012, Helsinki, Finland)*, pages 289–300, Berlin/Heidelberg, Germany. Springer-Verlag.

Rossum, G. V. (1993). Python for unix/c programmers copyright 1993 guido van rossum 1. In *Proc. of the NLUUG najaarsconferentie. Dutch UNIX users group*.

Torre, E., Picado-Muino, D., Denker, M., Borgelt, C., and Grün, S. (2013). Statistical evaluation of synchronous spike patterns extracted by frequent item set mining. *Frontiers in Computational Neuroscience*, 7.

Tsourakakis, C., Bonchi, F., Gionis, A., Gullo, F., and Tsiarli, M. (2013). Denser than the densest subgraph: Extracting optimal quasi-cliques with quality guarantees. In *Proc. 19th ACM SIGMOD Int. Conf. on Knowledge Discovery and Data Mining (KDD 2013, Chicago, IL)*, pages 104–112, New York, NY, USA. ACM Press.

Zaki, M. J., Parthasarathy, S., Ogihara, M., and Li., W. (1997). New algorithms for fast discovery of association rules. In *Proc. 3rd Int. Conf. on Knowledge Discovery and Data Mining (KDD 1997, Newport Beach, CA)*, pages 283–296, Menlo Park, CA, USA. AAAI Press.