Mining Frequent Synchronous Patterns with a Graded Notion of Synchrony

Salatiel Ezennaya-Gomez[1] **Christian Borgelt**[1]

[1]Intelligent Data Analysis Research Unit, European Centre for Soft Computing
c/ Gonzalo Gutiérrez Quirós s/n, 33600 Mieres (Asturias), Spain

Abstract

We present methods to find (significant) frequent synchronous patterns in event sequences, 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 constituting events. Since transferring earlier work (using a binary notion of synchrony) poses certain problems, we opt for an efficient approximation scheme to compute the pattern support. Furthermore, we transfer methods for filtering for significant and removing induced patterns, which require adaptations. Finally, we demonstrate the effectiveness of our approach with experiments on a large number of data sets with injected synchronous patterns.

Keywords: Keywords: graded synchrony, synchronous events, frequent pattern, pattern mining

1. Introduction

We present a methodology and algorithms to identify (significant) frequent synchronous patterns in event sequences, using the principles of frequent item set mining (FIM). The objective of FIM is to find all item sets that are frequent in a database, where an item set is called frequent if its support exceeds a (user-specified) minimum support threshold. While in standard FIM the support of an item set is simply the number of transactions it is contained in, the (usually) continuous time domain underlying event sequence data causes certain problems.

Earlier work in this area either relied on (naive) time binning to reduce the problem to the transactional case [13, 15] or defined that a group of items (event types) co-occur if their occurrence times are no farther apart than a (user-defined) maximum (time) distance. As a support measure, the latter approach uses a maximum independent set of such pattern instances as the support measure [4, 14].

Although this approach solves most of the problems of a time binning approach (specifically, the boundary problem, cf. [12]), it still suffers from some shortcomings. The shortcoming we focus on in this paper is that a binary notion of synchrony (that is, a group of events is either synchronous, namely if the events all occur within a certain limited time span, or not, namely if they occur farther apart) may not be so well suited for some applications. For example, we may desire a support measure that takes the *precision of synchrony* into account, so that a pattern that has fewer instances (i.e. 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 of these the synchrony of the events is rather loose.

This is the case in the application area that motivated our investigation, namely 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 [7] 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, the precision of synchrony is relevant, because synchronous spike input to receiving neurons is known to be more effective in generating output spikes [1, 9].

Therefore we propose in this paper, drawing on previous work [12], a graded notion of synchrony that gives rise to a support measure that captures both the number of instances as well as the precision of synchrony. Unfortunately, though, the efficient support computation algorithm proposed in [4, 14] for a binary notion of synchrony (that is, a greedy algorithm that finds a maximum independent set of the pattern instances) does not guarantee to produce the optimal result for this graded synchrony. In order to avoid having to apply a general maximum independent set algorithm (which has exponential time complexity, since the maximum independent set problem is NP-complete [8] and even hard to approximate [6]), we opt for an approximation that allows us to compute the support of a pattern very efficiently by intersecting interval lists.

Furthermore, we demonstrate how the tools to filter for statistically significant patterns (namely by analyzing surrogate data sets and employing pattern spectrum filtering [13, 15]) and to remove induced patterns (with pattern set reduction [15]) can be transferred from an approach based on binary

synchrony to our approach. Finally, we demonstrate the effectiveness of our procedure with experiments on a large number of data sets into which frequent synchronous patterns were injected.

The remainder of this paper is structured as follows: Section 2 covers basic terminology and notation and introduces our graded notion of synchrony. In Section 3 we show how the resulting support is approximated and frequent synchronous patterns are mined. In Sections 4 and 5 we show how pattern spectrum filtering and pattern set reduction, respectively, can be transferred from the binary case. 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 & Synchrony

Throughout this paper we adopt notation and terminology from [11]. Our 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. Note also that 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.

We define a *synchronous pattern* (in $\mathcal{S}$) 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. Hence the set of all instances of a pattern $I \subseteq B$, $I \neq \emptyset$, in an event sequence $\mathcal{S}$ is

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

where σ_w is the synchrony operator which measures the (degree of) synchrony of the events in $\mathcal{R}$. It may be chosen as (binary) synchrony [4, 14] defining

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

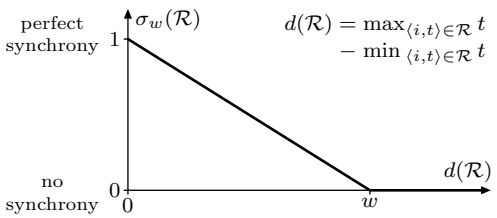

Figure 1: Degree of synchrony as a function of the distance between the latest and the earliest event.

However, here we are interested in a synchrony operator that yields a *degree of synchrony* between 0 and 1. Naturally, this operator should coincide with $\sigma_w^{(\mathrm{b})}$ for the limiting cases: if all events in $\mathcal{R}$ coincide (i.e. have the same occurrence 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 [12] 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

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

Note that an influence map is *not* a distribution function in the sense of probability theory, even though it shares its formal properties. In particular, it is *not* meant to describe uncertainty about the occurrence time of an event.

Based on influence maps, we define that there is synchrony to some degree between events iff the influence maps of these events overlap. The area of the overlap measures the degree of synchrony:

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

Alternatively, we may use the equivalent definition

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

This synchrony operator is illustrated in Figure 1.

The synchrony operator underlies the definition of a support operator $s_{\mathcal{S},w}(I)$ that we use to mine synchronous patterns. Intuitively, 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 [3].

The most natural support definition would be $s_{\mathcal{S},w}(I) = \sum_{\mathcal{R}\in\mathcal{E}_{\mathcal{S},w}(I)} \sigma_w(\mathcal{R})$, (which is equivalent to $s_{\mathcal{S},w}(I) = |\mathcal{E}_{\mathcal{S},w}(I)|$ for binary synchrony). However, this support measure is not anti-monotone [4]. Therefore we need to restrict the elements of $\mathcal{E}_{\mathcal{S},w}(I)$ over which we sum. Since distinct instances that share events cause the lack of anti-monotonicity, a fairly natural approach is that of two overlapping instances (that is, $\mathcal{R}_1, \mathcal{R}_2 \subseteq \mathcal{E}_{\mathcal{S},w}(I)$ with $\mathcal{R}_1 \neq \mathcal{R}_2$ and $\mathcal{R}_1 \cap \mathcal{R}_2 \neq \emptyset$) at most one should contribute to the support. This leads to a *maximum (size) independent set approach* (MIS) for binary synchrony and a *maximum (weight or degree of synchrony) independent set approach* for graded support [4, 12].

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

3. Support Computation & Pattern Mining

Our support operator, as defined in the preceding section, requires to compute a maximum (weight) independent set of $\mathcal{E}_{\mathcal{S},w}(I)$. Unfortunately, finding a maximum independent set is NP-complete in the general case [8] and even hard to approximate [6]. Intuitively, this means that (unless P = NP) there is no algorithm that does fundamentally better than trying all possibilities and thus needs exponential time (in the size of $\mathcal{E}_{\mathcal{S},w}(I)$). For binary synchrony, however, problem instances are strongly constrained by the underlying one-dimensional time domain. As a consequence, there exists an efficient greedy algorithm that yields an optimal result [4, 14]: If $\mathcal{S}$ is an event sequence over an item base B, $I \subseteq B$ an item set and w a (maximal) window width, then a maximum independent subset of $\mathcal{E}_{\mathcal{S},w}(I)$ can be found with a greedy algorithm, which always selects as the next instance the one consisting of the earliest event for each of the items in I that form an instance of I and are not contained in any already selected instance. A proof of the correctness of this procedure can be found in [14], pseudo-code in [4].

Unfortunately, applying the same greedy algorithm as for binary synchrony does not guarantee to find the optimal solution for support based on graded synchrony. Whether an alternative efficient algorithm exists for graded synchrony (since the problem instances are constrained as well, although less severely as with binary synchrony) we do not know. Hence, unless we want to execute a general MIS algorithm with exponential time complexity (as in [12]), we have to accept an approximation.

As such an approximation (Figure 2) we choose the integral over the maximum (union) of the minimum (intersection) of influence regions: the minimum represents the synchrony operator, the maximum aggregates over different instances. Formally:

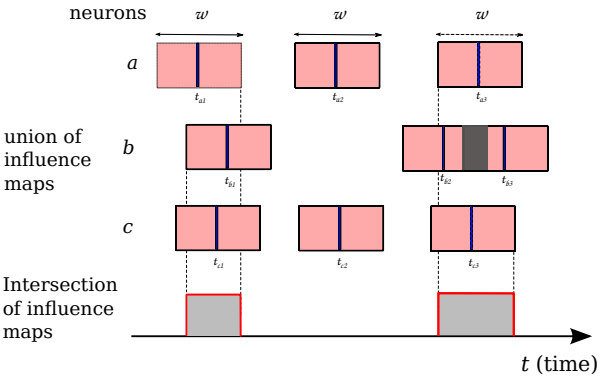

Figure 2: Support computation for three items a, b, c. Each event has its influence map (represented as a rectangle). If two influence maps overlap, the resulting influence map is the maximum (union) of these influence maps. The intersection of influence maps is the minimum which defines the synchrony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

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

Exploiting the properties of maxima and minima, this definition can conveniently be rewritten as

$$s_{\mathcal{S},w}(I) = \int_{-\infty}^{\infty} \min_{i\in I} \Big(\max_{\langle j,t\rangle\in\mathcal{S};j=i} f_t(x) \Big) \,\mathrm{d}x.$$

This form of the support measure has (at least) two advantages: in the first place, it is clearly anti-monotone (this is obvious from the minimum over $i \in I$). The second is that it allows to compute support by a simple intersection of interval lists, since all occurring functions only take two values, namely 0 and $\frac{1}{w}$, and thus it suffices to record where they are greater than 0. Hence, as a preprocessing step, we compute for each item $i \in B$ the list of intervals in which $\max_{\langle j,t\rangle\in\mathcal{S};j=i} f_t(x) > 0$. These intervals can then be intersected to account for the minimum. Finally, the area under the functions is obtained by summing the interval lengths and dividing by w.

Note that this computation can be seen as a natural generalization of the transaction list intersection carried out by the *Eclat algorithm* [16] to a continuous domain. As a consequence, Eclat's item set enumeration scheme, which is based on a *divide-and-conquer* approach [3], can be transferred with only few adaptations to obtain an efficient algorithm for mining frequent synchronous patterns. The divide-and-conquer scheme can be roughly characterized as follows: for a chosen item i, the problem to find all frequent patterns is split into two subproblems: (1) find all frequent patterns containing item i and (2) find all frequent patterns *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.

In order to reduce the output we restrict it to closed frequent patterns. A pattern is called *closed* if no super-pattern has the same support. Closed patterns have the advantage that they preserve knowledge of what patterns are frequent and allow us to compute the support of any non-closed frequent pattern easily (see [3]). However, it should be noted that the restriction to closed patterns is less effective with graded synchrony than with binary synchrony, because adding an item can now reduce the support not only by losing instances, but also by worsening the precision of synchrony. Hence, most patterns are closed under graded synchrony.

4. Pattern Spectrum Filtering

The output of pure synchronous pattern mining is usually (much) too large to be useful and thus further reduction is necessary. One way of doing this is to identify statistically significant patterns. Previous work showed that statistical tests on individual patterns are not suitable [13, 15] (even though these papers considered time-binned data, their arguments apply to our setting as well). 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. [5].

To overcome this problem, we rely here on the approach suggested in [13] and refined in [15], 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. 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 if 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.

From this insight it was derived in [13] 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. It is claimed that a pattern cannot be called significant if a counterpart (that is, 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 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 [10].

The only adaptation that we have to make compared to [13, 15] is that, due to our graded synchrony, support values are no longer integers, but can be any (non-negative) real number. As a consequence, we have to change the pattern spectrum from a bar chart with discrete values on both axes to a histogram, where support bins with a (user-specified) width are formed for the support axis. An example of such a pattern spectrum, for data as it will be used in Section 6, is shown in Figure 5(a). It captures what pattern signatures occurred in a large number of surrogate data sets.

5. Pattern Set Reduction

Unfortunately, even after pattern spectrum filtering, many spurious patterns may remain. Such patterns are caused by an actual pattern interacting with background chance events, which gives rise to subset, superset and overlap patterns. Supersets result from items outside of an actual pattern occurring by chance together with some of the instances of the actual pattern. Subsets result from some of the items in the actual pattern occurring together, in addition to the instances of the actual pattern, as a chance event. Finally, overlap pattern result from items outside of an actual pattern co-occurring with some instances of the injected pattern and at least one chance event of a subset of the actual pattern.

In order to remove such spurious induced patterns, we draw on *pattern set reduction* (PSR), as it was proposed in [15] for time-binned data, and transfer it to graded synchrony. The basic idea of pattern set reduction is to define a *preference relation* between patterns X and Y with $Y \subset X \subseteq B$. Only patterns to which no other pattern is preferred are kept. All other patterns are deleted.

The preference relations considered in [15] are based on three core principles: (1) explaining *excess coincidences* (subsets), (2) explaining *excess items* (supersets) and (3) assessing the pattern probability based on the *number of covered events*. More formally, let $z_X = |X|$ and $z_Y = |Y|$ be the sizes of the patterns X and Y, respectively, and let $s_X = s_{\mathcal{S},w}(X)$ and $s_Y = s_{\mathcal{S},w}(Y)$ be their support values. Since we have $Y \subset X$ it follows $z_X > z_Y$ and $s_Y > s_X$, because support is anti-monotone and we consider only closed patterns. With (1), X

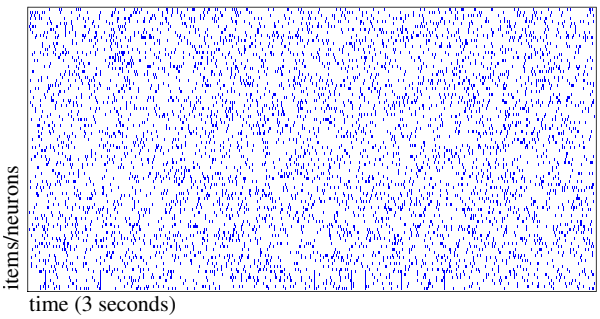

Figure 3: Example of generated data sets that imitate parallel neural spike trains.

is preferred to Y if the excess coincidences $s_Y - s_X$ that Y exhibits over X can be explained as a chance event. With (2), Y is preferred to X if the presence of $z_X - z_Y$ excess items that X contains over Y can be explained as a chance event. With (3), the pattern is preferred for which $z \cdot s$ (or, alternatively, $(z-1) \cdot s$) is larger. In the case of time-binned data or binary synchrony, $z \cdot s$ is the number of individual events supporting a pattern. In the alternative version, the events of a reference item, to which the events of the other items are synchronous (as it makes no sense to speak of synchronous events if there is only one item), are disregarded.

Transferring (1) and (2) to the case of graded synchrony turns out to be difficult, because the decision whether excess coincidences or excess items can be explained as chance events is made based on the pattern spectrum, using heuristic signature modifications that are tricky to transfer to a non-integer support. In addition, they are based on pairwise pattern comparisons, while the third approach relies on a potential function (in the sense of physics), thus simplifying the reduction process. As a consequence, we focus here on the third method, even though its intuitive justification as a number of events is also lost. However, we argue that the product of size (or size minus 1) and support can also be justified as derived from the shape of the decision border between significant and non-significant patterns as it is induced by the pattern spectrum. (This border generally has a hyperbolic shape and is derived, in the binary case, from curves of pattern signatures having equal probability under certain simplifying assumptions—see [15].)

However, the graded synchrony we employ in this paper forces us to adapt this function. The reason is that with graded synchrony increasing the pattern size generally reduces the support, and not necessarily because instances get lost, but because the precision of synchrony is reduced by added items. This needs to be taken into account in the evaluation function. Since the loss of synchrony depends on the number of items, we heuristically chose $(z-1)(s+kz)$, where k is a (user-specified) parameter that is meant to capture the loss of synchrony relative to the pattern size.

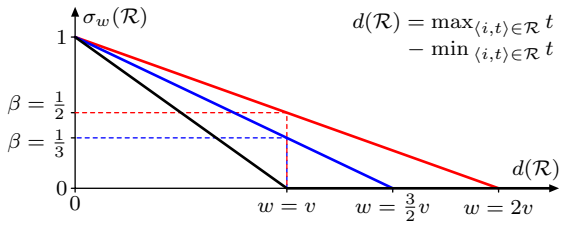

Figure 4: Effect of increasing the window w beyond the expected amount v of jitter in the data.

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 pattern spectrum estimation algorithms (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).

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 $[-\frac{v}{2}\text{ms}, +\frac{v}{2}\text{ms}]$ with $v = 2, 3, 4, 5\text{ms}$ (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 3.

Then we tried to detect the injected synchronous patterns with the methods described above, first carrying out pattern mining on the original data (with a minimum support $s_{\min} = 1$ and a minimum pattern size $z_{\min} = 2$), then filtering it with a pattern spectrum derived from 10,000 data sets with independent spike trains (as shown in Figure 5(a)), and finally applying pattern set reduction. Apart from the jitter width v, we varied two parameters: the window width w and the parameter k (pattern set reduction, see Section 5). The parameter k was varied between 0.1 and 0.2, in steps of 0.01.

The reason for varying w is as follows: while with binary synchrony it is obvious that we should choose $w = v$ (since this allows us to capture all pattern instances, while incurring a minimum of chance patterns), using the same relation is not likely to produce the best results for graded synchrony. This becomes obvious if we consider the extreme case that all instances of a pattern realize the maximum spread v (that is, for all instances the earliest and the latest event are v apart). In this case the support of the pattern is 0, regardless of the number of

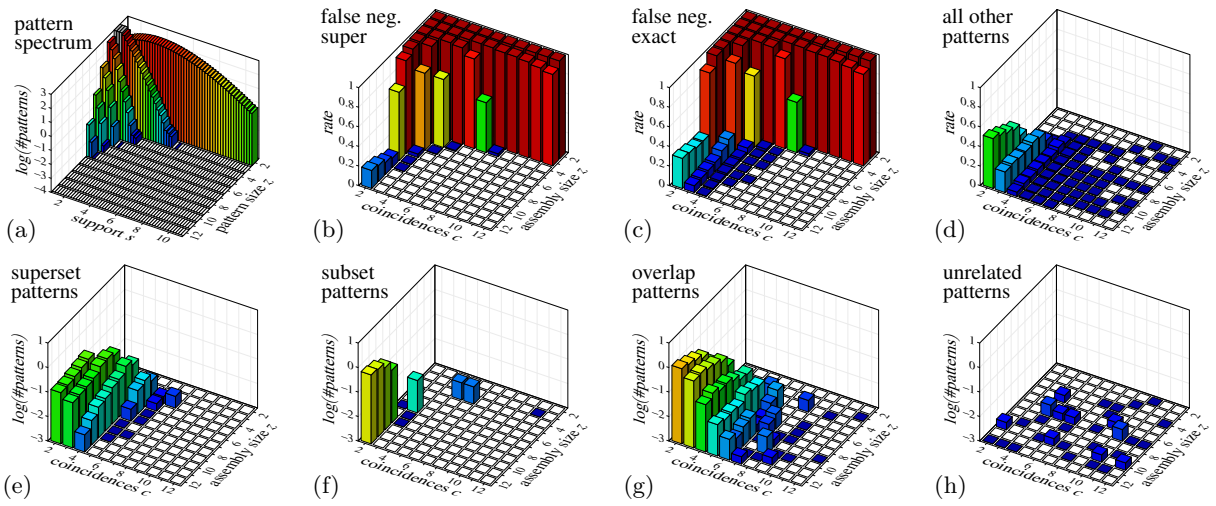

Figure 5: Results for injected synchronous patterns with signatures $(z, c) \in \{2, \ldots, 12\}^2$, instances jittered with $v = 2$ms. Pattern mining executed with $w = 3.6$ms windows and reduced with $(z-1)(c+kz)$ where $k = 0.15$. Individual diagrams: (a) pattern spectrum derived from 10,000 surrogate data sets; (b) false negatives rates if supersets of injected pattern count as detections; (c) false negative rates for exact detections; (d) rate of patterns other than the injected patterns (superset, subsets, overlap patterns unrelated patterns); (e) number of reported supersets of injected pattern; (f) number of reported subsets of injected pattern; (g) number of reported patterns that overlap the injected pattern; (h) number of patterns that are unrelated to injected pattern (at most one item in common). Diagrams (b)–(h) are derived from/averaged over 1000 data sets. Note the logarithmic scale on the vertical axis in diagrams (e)–(h).

instances, because the degree of synchrony of each individual instance is 0. That is, with $w = v$ we have, in a way, gone too far with graded synchrony: while we wanted only to introduce a (partial) dependence on the precision of synchrony, we achieved that the precision of synchrony submerges the dependence on the number of instances.

Fortunately, though, this can easily be fixed by choosing a window width $w > v$, say $w = \alpha \cdot v$. Note that even though the value v of the jitter width is generally unknown in practice, there usually exists some expectation of its size. Choosing $w = \alpha \cdot v$ then merely means that one has to multiply this expectation by a certain factor. Note also, that the same problem exists for binary synchrony, where we merely know that $\alpha = 1$ is optimal.

The effect of choosing a value $w > v$ is demonstrated in Figure 4: while with $w = v$ the entire degree of synchrony depends on the precision of synchrony, $w = \alpha \cdot v$ yields a minimum degree of synchrony of $\beta = \frac{\alpha-1}{\alpha}$ (provided that the jitter of patterns in the data is actually limited to v), while only $1 - \beta = 1 - \frac{\alpha-1}{\alpha} = \frac{1}{\alpha}$ depends on the precision of synchrony. Hence, with this approach, the support consists of two elements: an element that derives from the number of instances (which is controlled by $\beta = \frac{\alpha-1}{\alpha}$) and an element that captures the precision of synchrony (which is controlled by $1 - \beta = \frac{1}{\alpha}$). In our experiments, we varied the factor α between 1 and 2, in steps of 0.1.

Example results for a choice of the analysis parameters (window width w and parameter k) that seemed optimal after comparing to alternative settings are shown in Figure 5. False negatives occur mainly for small pattern sizes and few coincidences, which is mainly due to the fact that such patterns can be explained as chance events (that is, the patterns are deleted by pattern spectrum filtering). However, for sufficiently large patterns (four and more items) and sufficiently many instances (five to six are almost always sufficient), the detection is essentially perfect. Alternative results, especially for different factors $\alpha = \frac{w}{v}$ and different values of the parameter k, are shown in Figure 6. We see that choices different from the one in Figure 5 deteriorate the detection quality.

7. Conclusions

In this paper we presented a method for mining frequent synchronous patterns in event sequences based on a graded notion of synchrony. In order to avoid having to solve a maximum independent set problem, we chose an approximation of the induced support measure, which is efficiently computable with interval list intersection. We transferred the methods of pattern spectrum filtering and pattern set reduction, adapting the filtering process and the pattern evaluation function, so that they take care of the underlying graded synchrony measure. Our experiments demonstrate that the whole process, consisting of pattern mining, pattern spectrum filtering and pattern set reduction works very well and can detect even relatively small patterns with fairly few instances. Since we are defining and improving a *detection method* we worked with artificial data,

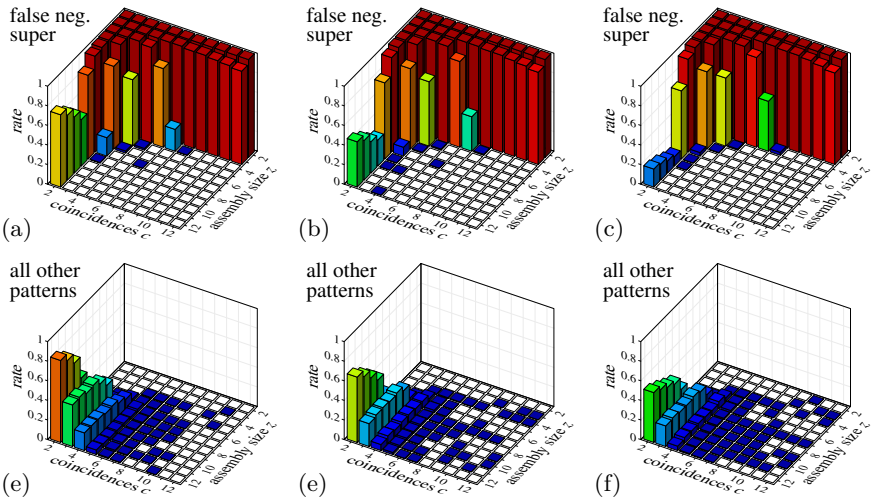

Figure 6: Results for injected patterns with signatures $(z, c) \in \{2, \ldots, 12\}^2$, instances jittered with $v = 2$ms. Pattern set reduction executed with $(z-1)(c+kz)$. (a) false negatives (FN) for $w = 3.2$ms and $k = 0.2$; (b) FN for $w = 3.4$ms and $k = 0.16$; (c) FN for $w = 3.6$ms and $k = 0.15$; diagrams (d), (e) and (f) correspond to (a), (b) and (c), respectively, and show the rate of other patterns.

for which we know whether it contains what we want to detect (which is impossible for real world data). In the future, we plan to investigate possible pattern set reduction approaches more thoroughly and extend our method to handle selective participation, i.e. the occurrence of incomplete instances. Finally we plan to apply our method to real world data.

Software and Additional Results

Python and C implementations of synchronous pattern mining with binary synchrony can be found at:
http://www.borgelt.net/coconad.html and
http://www.borgelt.net/pycoco.html.
Modified versions, which use a graded notion of synchrony and which we developed for the experiments in this paper, will soon be made available at the same URLs. Extended result diagrams as well as the Python scripts with which we conducted out experiments will be made available at:
http://www.borgelt.net/ovlexp.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] M. Abeles. Role of the Cortical Neuron: Integrator or Coincidence Detector? *Israel Journal of Medical Sciences* 18(1):83–92. Israel Medical Association, Ramat Gan, Israel 1982

[2] R. Bhandari, S. Negi, and F. Solzbacher. Wafer Scale Fabrication of Penetrating Neural Electrode Arrays. *Biomedical Microdevices* 12(5):797–807. Springer, New York, NY, USA 2010

[3] 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

[4] 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

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

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

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

[8] 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

[9] P. König, A.K. Engel, and W. Singer. Integrator or Coincidence Detector? The Role of the Cortical Neuron Revisited. *Trends in Neurosciences* 19(4):130–137. Cell Press, Maryland Heights, MO, USA 1996

[10] 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

[11] 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

[12] D. Picado-Muiño, I. Castro-León, and C. Borgelt. Fuzzy Frequent Pattern Mining in Spike Trains. *Proc. 11th Int. Symposium on Intelligent Data Analysis (IDA 2012,*

Helsinki, Finland), 289–300. Springer-Verlag, Berlin/Heidelberg, Germany 2012

[13] 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

[14] 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

[15] 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

[16] 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