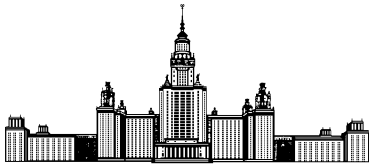
Московский государственный университет имени М. В. Ломоносова



Факультет Вычислительной Математики и Кибернетики

Кафедра Математических Методов Прогнозирования

ДИПЛОМНАЯ РАБОТА СТУДЕНТА 517 ГРУППЫ

«Параллельная реализация метода поиска закономерностей в последовательностях событий»

Выполнил:

студент 5 курса 517 группы

Вишневский Валерий Викторович

Научный руководитель:

к.ф-м.н., н.с.

Ветров Дмитрий Петрович

Заведующий кафедрой

Математических Методов

Прогнозирования, академик РАН _____ Ю. И. Журавлёв

К защите допускаю

«_____» _____ 2010 г.

К защите рекомендую

«_____» _____ 2010 г.

Москва, 2011

Аннотация

В данной работе предлагается оригинальный алгоритм для поиска скрытых закономерностей в последовательностях событий, основанный на вероятностном представлении паттернов(закономерностей) в дискретных последовательностях событий. Также рассматривается применение данного алгоритма для анализа поведения мышей. В результате чего, по полученному множеству паттернов можно определить группу мышей, к которой относится наблюдаемая особь.

Проведено сравнение реализованного алгоритма с существующими аналогами, показавшее, что предложенный метод более устойчив к шуму в исходных данных и позволяет найти более значимые закономерности. Приведен анализ параметров алгоритма. Предложенный метод поиска паттернов основывается на определении взаимосвязи между парами событий. Поиск производится снизу вверх: алгоритм сначала находит простые закономерности, потом, путем соединения простых, образуются более сложные паттерны. На каждом шаге проводится отбор самых существенных и полных паттернов.

Данный алгоритм был реализован в среде `Matlab`. Наиболее вычислительно-сложные процедуры были реализованы в виде `mex`-файлов на языке `C++`, с использованием технологии `CUDA`. Параллельная реализация отдельных процедур на `GPU` дала ускорение в 30 и 140 раз, по сравнению с последовательной версией на языке `C++`.

Содержание

1 Введение

Задача поиска закономерностей(стереотипов, паттернов, шаблонов — здесь синонимы) в поведении животных и людей крайне важна в современной нейробиологии и когнитивных науках. Выделив характерные паттерны, мы, например, можем делать выводы о сложности поведения различных особей, определять изменения в поведении наблюдаемых процессов, другими словами, решив задачу поиска паттернов, мы можем определенным образом *измерять* поведение особи, или группы особей, становится возможно более наглядно анализировать поведение. Именно анализ поведения является основным инструментом при исследовании на системном уровне механизмов работы памяти и обучения животных.

В современной нейробиологии можно выделить три основных подхода к описанию поведения [1, с. 57].

Структурное описание: определяет какие двигательные действия животное совершает в каждый момент времени. Например, для животного-робота, такое описание будет служить последовательностью инструкций, выполняя которые можно воспроизвести какое-либо поведение.

Эффектное описание: определяет, какие эффекты имеет текущее поведение на само животное, или окружающую его среду, причем сами физические действия субъекта, приведшие к определенному событию обычно не рассматриваются. При таком подходе, поведения может описываться, например, следующими *терминами*: «кормление», «проявление агрессии», «сбор нектара», «уход от хищника».

Пространственное описание: в данном случае, поведение животного описывается траекторией его движения во времени. Это описание может быть усложнено, например, добавлением информации об ориентации животного, или о том, в каком темпе производится движение(украдка, бег, и т. д.). В то же время, пространственное описание может быть упрощено, например, клетка разбивается на несколько зон, после чего, подсчитывается время проведенное животным в каждой из зон.

Например, «включить свет» — это действие в терминах эффектного описания, будет описано, как «нажать на выключатель указательным пальцем» в терминах структурного описания.

Все три метода, по сути, описывают поведение, как некоторый процесс во времени. Можно заметить, что в первом и втором случае поведение описывается «более дискретно», поэтому обычно методы, применяемые для исследования пространственного описания поведения, неприменимы для исследования структурного и эффектного описания, а для исследования структурного и эффектного описания обычно используются одинаковые подходы.

Мы будем рассматривать случай, когда поведение описано в структурных или эффектных терминах. Тогда, в каждый момент времени может происходить смена состояния, подразумевается, что начало текущего поведенческого акта совпадает с концом предыдущего. Так мы можем записывать только моменты времени, когда поведенческий акт *начинается*. Множество документируемых поведенческих актов определяется экспертами на этапе планирования эксперимента, а сама разметка поведения на эти акты может производится по данным видеонаблюдения вручную, или автоматически с помощью специализированного ПО, например, `Noldus Observer`.

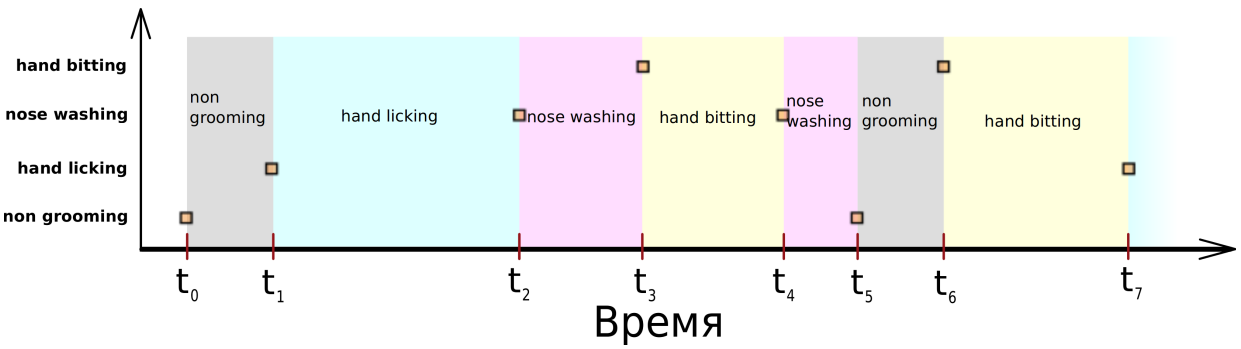


Рис. 1: Пример описания поведения. По вертикали отложены фиксируемые поведенческие акты, по горизонтали — время. Начало каждого следующего акта является концом предыдущего.

Многие паттерны поведения являются тривиальными и понятными. В человеческом поведении, большинство повседневных церемоний, ритуалов приветствия, рабочих и игровых процессов, по сути, являются поведенческими паттернами. Примером повседневного паттерна может быть процесс принятия пищи: «сесть за стол», «съесть

главное блюдо», «съесть десерт», «выпить чай», «встать из-за стола». В свою очередь, акт «сесть за стол» может являться составным действием, состоящим из следующих актов: «выдвинуть стул», «передвинуть тело к стулу», «согнуть колени, чтобы опустить торс». Таким образом, мы видим, что поведенческий паттерн обладает иерархией: более простые события могут быть объединены в простые паттерны(здесь *подпаттерны*), которые, в свою очередь, объединяются в более сложные паттерны. Важно помнить, что сами события и подпаттерны могут возникать отдельно, не в составе более сложных паттернов.

Однако в основном, в виду большого объема данных и сложной структуры поведения, паттерны очень сложно выделить визуально. Из-за этого такие паттерн иногда называют скрытыми(hidden). Именно для таких случаев требуется разработка отдельных методов. Для каждого эксперимента стандартное количество документируемых актов — десятки, количество появлений каждого акта — сотни.

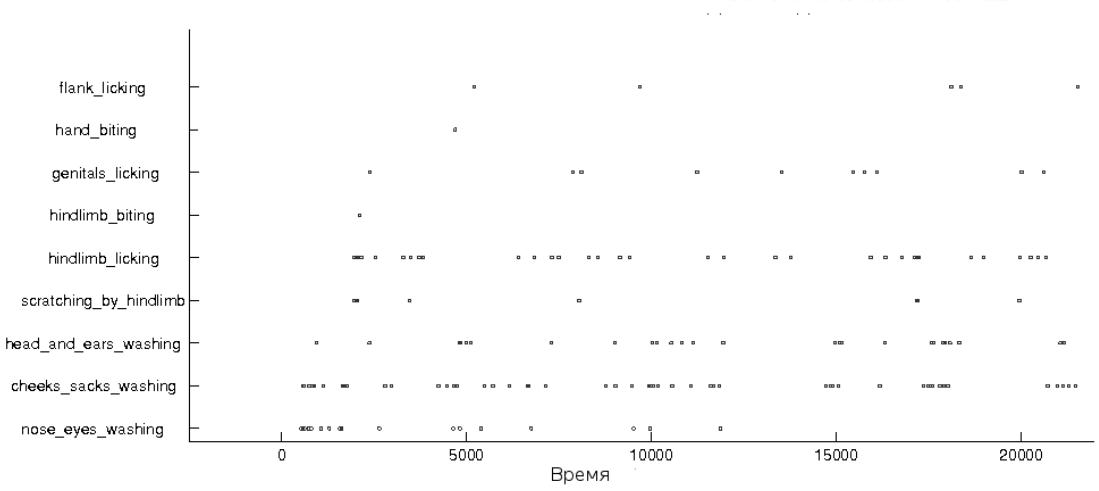


Рис. 2: Пример реальных поведенческих данных.

Приведенные выше рассуждения дают понятие о том, что такое поведенческий паттерн, как его можно моделировать формальными терминами (более точно, здесь дано определение Т-Паттерна(T-Pattern), введенное Магнусом Магнуссоном в [2]). Иерархичная структура паттерна задает основы стратегии поиска подобных паттернов.

1.1 Определения и обозначения

Пусть время наблюдения разбито на N_t интервалов. В каждый момент *периода наблюдения* $[1, N_t]$ может произойти некоторое событие $\mathbf{e}$(*действие, поведенческий акт, event*) [1] из множества допустимых событий $\mathcal{E}$ (*event types*). Соответственно, каждому типу события сопоставляется множество моментов времени $TS(\mathbf{e})$:

$$TS(\mathbf{e}) = \{t^{\mathbf{e}}_1, \ldots, t^{\mathbf{e}}_{N_\mathbf{e}}\}, \qquad \mathbf{e} \in \mathcal{E}, \qquad 0 \leqslant t^{\mathbf{e}}_i \leqslant N_t, \quad (i = 1, \ldots, N_t),$$

здесь $N_\mathbf{e}$ — количество появлений события $\mathbf{e}$ в данных.

Нечетким паттерном(далее P-Pattern, или просто паттерн) $\mathbf{P}$ длины $N_\mathbf{P}$ назовем упорядоченную последовательность событий $\mathbf{e}_i, (i = 1, \ldots N_\mathbf{P})$, где каждое событие паттерна характеризуется смещением и разбросом от предыдущего события. Будем записывать паттерн $\mathbf{P}$ в следующем виде:

$$\mathbf{P} = \mathbf{e}_1[\mu_1, \sigma_1]\mathbf{e}_2[\mu_2, \sigma_2] \ldots \mathbf{e}_{N_\mathbf{P}}[\mu_{N_\mathbf{P}}, \sigma_{N_\mathbf{P}}], \qquad \mu_1 = 0.$$

Здесь μ_i и σ_i — смещение и разброс соответствующего события относительно предыдущего(см. Рис. 3). Распределение межточечных расстояний [6, с. 138] между событиями в паттерне моделируется нормальным распределением. Если нам не важны параметры смещения разброса событий, то паттерн записываться так: $\mathbf{P} = \mathbf{e}_1\mathbf{e}_2 \ldots \mathbf{e}_{N_\mathbf{P}}$, если нам важна иерархия паттерна, то например так: $\mathbf{P} = ((\mathbf{e}_1\mathbf{e}_2)(\mathbf{e}_3\mathbf{e}_4))$

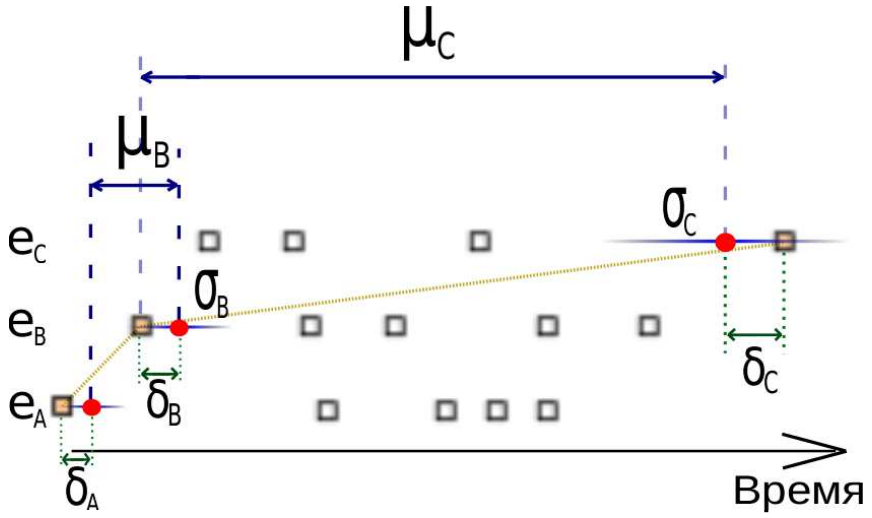


Рис. 3: Представление нечеткого паттерна $\mathbf{P} = \mathbf{e}_A[\mu_A, \sigma_A]\mathbf{e}_B[\mu_B, \sigma_B]\mathbf{e}_C[\mu_C, \sigma_C]$.

[1] Чаще всего понимается, что в этот момент времени имеет место *начало* действия

Далее, что бы иметь возможность обрабатывать пропуски в реализации паттернов, введем понятие *функции потерь*, которая определяет «штраф» за пропуск m событий в паттерне длины $N_{\mathbf{P}}$ следующим образом:

$$f_{LOSS}(m, N) = \begin{cases} \exp\left(-\frac{\lambda m}{N_{\mathbf{P}}}\right), & m < N, \\ 0, & m = N. \end{cases}$$

Здесь λ является структурным параметром, определяющим максимальный допустимый уровень «нечеткости» паттернов. Если этот параметр велик, то мы, по сути, запрещаем реализациям паттерна иметь пропуски. Если выставить этот параметр слишком малым, то будут обнаруживаться паттерны, не разу полностью не встречающиеся в данных, то есть закономерности могут быть найдены даже в случайных данных. Данный параметр должен выставляться вручную, исходя из априорной информации о типе данных, уровне шума, и сложности поведения.

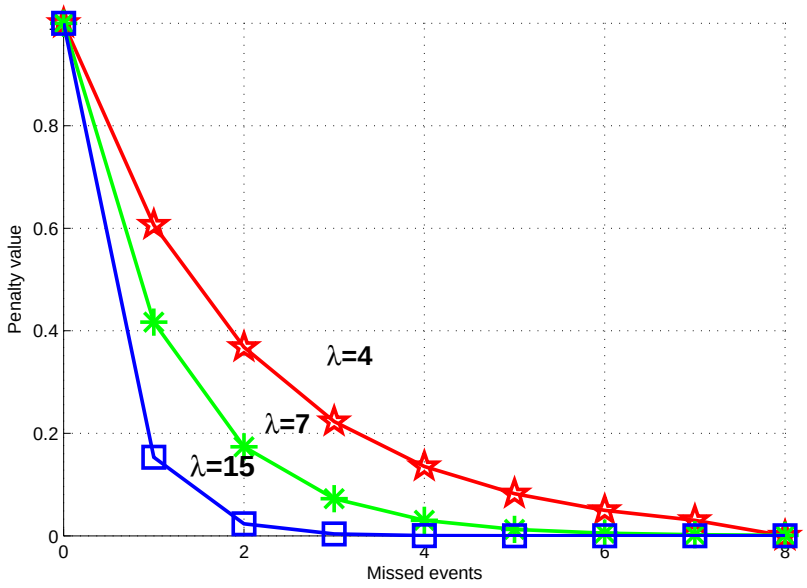


Рис. 4: Функция потерь для паттерна длины 8 при разных значениях параметра λ.

Теперь мы можем определить ключевое понятие для представленного метода — *правдоподобие паттерна*. Правдоподобие паттерна $\mathbf{P}$ — это функция, определенная для каждого момента времени наблюдения ε ($\varepsilon = 1, \ldots, N_t$), показывающая на сколько четко можно говорить, что данный паттерн начинается в данный момент времени ε. Более формально:

$$L_{\mathbf{P}}(\varepsilon) = f_{LOSS}(N_-, N_{\mathbf{P}}) \prod_{i=1}^{N_{\mathbf{P}}} \left(\frac{1}{\sqrt{2\pi}\,\sigma_i} \right) \prod_{i \in \mathcal{N}_+} \exp\left(-\frac{\delta_i^2}{2\sigma_i^2} \right), \tag{1}$$

здесь δ_i — расстояния от ожидаемой позиции события в паттерне до ближайшего события в данных(более наглядно см. Рис. 3). Т.е.:

$$\delta_i = \min_{x \in TS(\mathbf{e}_i)} \left| \underbrace{\varepsilon + \sum_{j=1}^{i-1} (\mu_j + \delta_j) + \mu_i}_{\text{ожидаемая позиция события}} - x \right|.$$

Далее, N_- — количество пропущенных событий в паттерне, а $\mathcal{N}_+$ — множество индексов присутствующих в паттерне событий. Событие считается пропущенным, если $\exp\left(-\frac{\delta_i^2}{2\sigma_i^2}\right) < \exp\left(-\frac{\lambda}{N_{\mathbf{P}}}\right)$, т.е. соответствующее значение δ_i больше определенного предела.

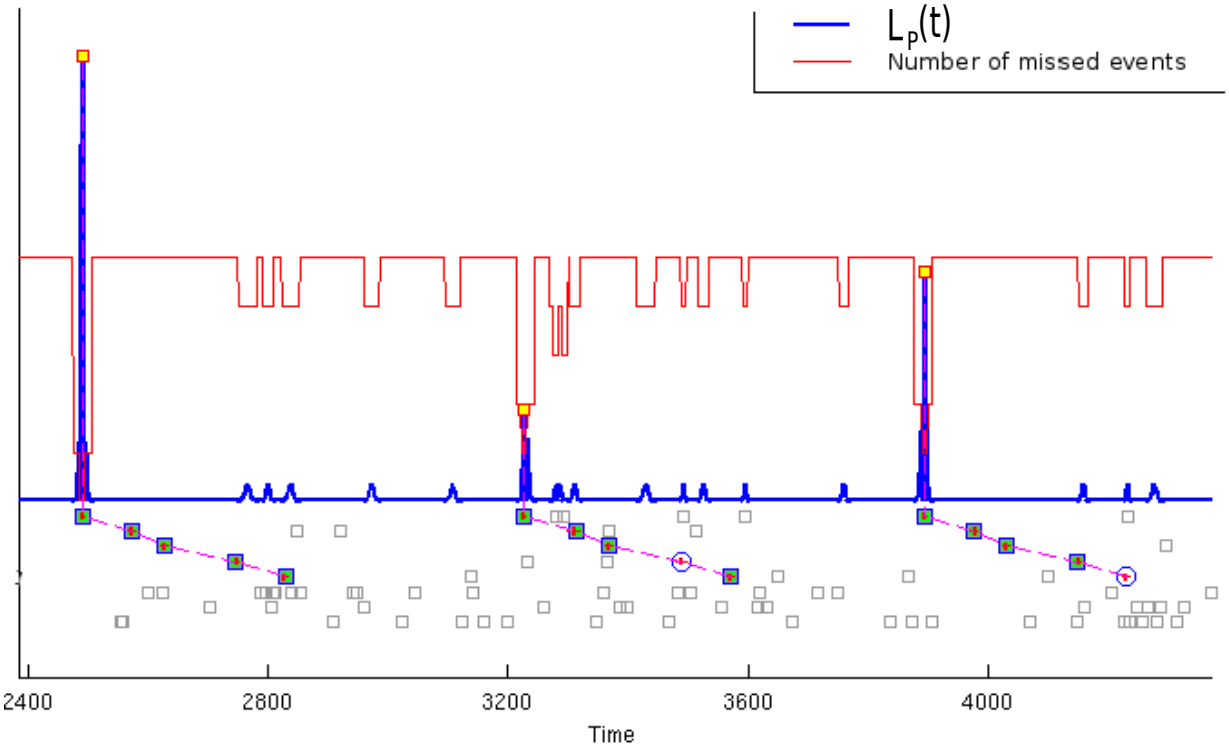


Рис. 5: Пример функции правдоподобия паттерна. Желтыми маркерами с красной границей изображены максимумы функции правдоподобия: моменты времени, когда мы считаем, что паттерн имеет место быть. Значение функции правдоподобия в максимуме будет весом $\alpha_{\mathbf{P},k}$ k-го появления паттерна $\mathbf{P}$ в данных. В нижней части рисунка закрашенными квадратами показаны присутствующие события, полыми кружками — пропущенные события в паттерне. Полые серые квадраты соответствуют наблюдаемым поведенческим актам.

Заметим, что правдоподобие паттерна можно считать не только начиная с первого события, но и, например, с конца паттерна. Для упрощения вычислений, значение правдоподобия паттерна $\mathbf{P}$ с события m можно рассчитывать по следующей формуле:

$$L_{P,m}(\varepsilon) = L_P\left(\varepsilon + \sum_{j=1}^{m} \mu_j\right).$$

На рисунке 6 видно, что правдоподобие, посчитанное с m-го события имеет максимумы в момент наступления m-го события в текущем паттерне. Данный факт будет активно использоваться при вычислении межточечных расстояний между паттернами и в процессе редукции паттернов-дубликатов.

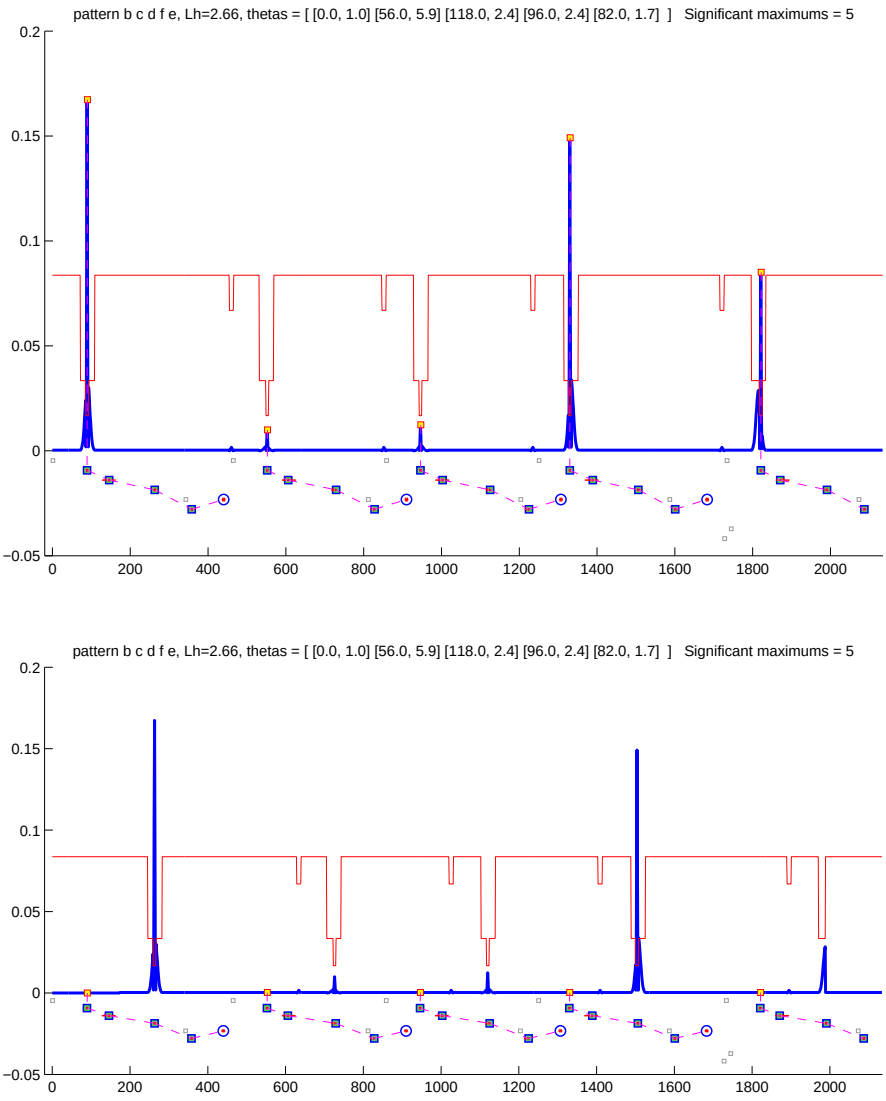


Рис. 6: Функция правдоподобия одного и того же паттерна, подсчитанная, начина с первого события $m = 1$(сверху) и с третьего события $m = 3$(снизу).

1.2 Обзор существующих методов поиска паттернов и закономерностей

Несмотря на то, что описанные выше паттерны широко распространены в описании поведения, стандартные статистические методы не подходят для их поиска [2]. Еще сложнее приспособить статистические программные пакеты для выявления таких закономерностей. Отчасти это объясняется тем, что для исследования более простых систем(не обязательно биологических), например, мух дрозофил, достаточно

оперировать более простыми статистическими данными о поведении. Закономерности в поведении таких систем можно моделировать, например, с помощью периодических орбит из теории символьной динамики [5], в таком случае паттерн определяется только последовательностью событий, но не временными интервалами, связывающими их. Так как поведение дрозофил достаточно примитивно(оно слабо варьируется, сильно предопределено внешними факторами и предыдущими состояниями), описание возникающих паттернов периодическими орбитами вполне уместно [4]. Однако для исследования поведения млекопитающих и, тем более, поведения человека, такие простые техники не могут описать в требуемом объеме всю сложность поведения. И именно временно́е описание паттерна призвано решить проблему представления и поиска поведенческих закономерностей в сложных системах.

Существует подход к поиску закономерностей в дискретных временных рядах, основанный на проходе по исходных данных окном заданной ширины и подсчетом определенных частотных характеристик в этом окне(например [9]). Но такой подход так же не удовлетворяет поставленным требованиям к искомым закономерностям.

Методы поиска закономерностей из смежных областей, как правило, нацелены на поиск совершенно других структур: это либо просто определенные тренды во временных рядах, либо ассоциативные правила, которые не передают временну́ю и иерархическую структуру закономерности. Подход к поиску паттернов в активности нейронных культур [7, 8], в виду того, что приходится оперировать с крайне большими объемами данных, основывается на определении метрики между группами событий и дальнейшей их кластеризации, также не подходит в случае анализа поведения животных.

1.3 Алгоритм поиска T-Паттернов

На сегодняшний день, для анализа временны́х поведенческих закономерностей наиболее широкое распространение получил метод поиска T-Паттернов, предложенный в 2000-ом году Магнусом Магнуссоном в [2](также см. в [10]). Так как предложенный в нашей работе метод является, по сути, расширением метода Магнуссона, опишем далее основные понятия алгоритма поиска T-Паттернов.

Паттерна вводится так же как, и в подразделе 1.1, за исключением того, что каждое событие паттерна определяется фиксированным временны́м интервалом, в течение которого это событие должно присутствовать после предыдущего события. Другими словами, здесь расстояния между событиями моделируется не нормальным распределением, а равномерным. Т-Паттерн обычно обозначается следующим образом:

$$\mathbf{P} = \mathbf{e}_1[\,d_L^1, d_R^1]\mathbf{e}_2[\,d_L^2, d_R^2]\ldots\mathbf{e}_{N_\mathbf{P}-1}[\,d_L^{N_\mathbf{P}-1}, d_R^{N_\mathbf{P}-1}]\mathbf{e}_{N_\mathbf{P}}.$$

Здесь запись $\mathbf{e}_A[\,d_L, d_R]\mathbf{e}_B$ обозначает, что событие $\mathbf{e}_B$ должно присутствовать во временно́м интервале $[\,d_L, d_R]$ после события $\mathbf{e}_A$, чаще, чем это ожидается при предположении о независимости событий. Более точно, при помощи биномиальной схемы, считается вероятность текущей конфигурации распределения событий, принимая гипотезу, о том, что вероятность встретить событие $\mathbf{e}$ в какой-либо момент времени равна $\frac{N_\mathbf{e}}{N_t}$. Если эта вероятность меньше заданного порога α(обычно берется меньше 0.05), то гипотеза о независимом равномерном распределении событий отвергается и мы говорим, что $\mathbf{e}_A[\,d_L, d_R]\mathbf{e}_B$ — Т-Паттерн, соединенные отношение критической связи(critical relation). Точно такие же рассуждения можно провести, взяв вместо событий $\mathbf{e}_A$ и $\mathbf{e}_B$ составные Т-Паттерны $\mathbf{P}_A$ и $\mathbf{P}_B$. Потребуется только считать вхождения начала Т-Паттерна $\mathbf{P}_B$ в интервале $[\,d_L, d_R]$ после конца Т-Паттерна $\mathbf{P}_A$.

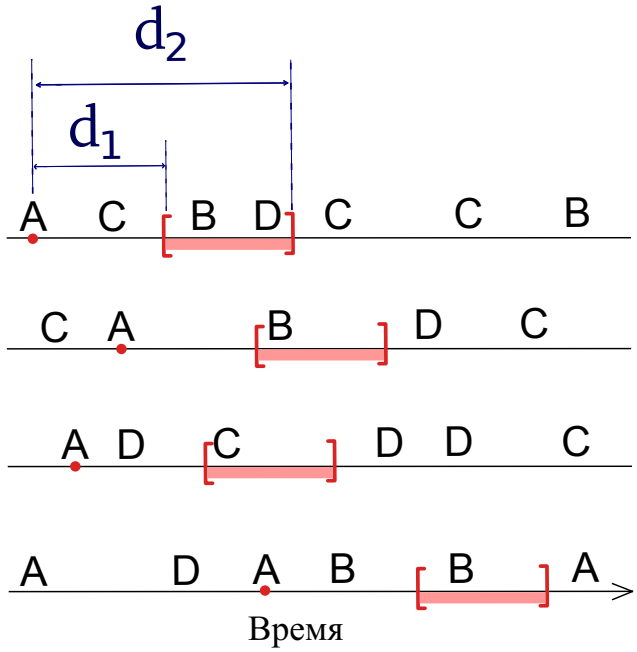


Рис. 7: Отношение критического интервала $[\,d_1, d_2]$ между событиями A и B.

Опишем этап *конструирования* Т-Паттернов: для всех пар Т-Паттернов, определяем, существует ли интервал $[d_L, d_R]$, связывающий эту пару паттернов критическим отношением. Если такой интервал существует, то мы добавляем новый Т-Паттерн. Если таких интервалов несколько, то предлагается взять самый длинный интервал.

Для очистки множества найденных паттернов от паттернов-дубликатов и неполных копий существующих паттернов, вводится процедура *редукции* множества Т-Паттернов: Т-Паттерн $\mathbf{Q_x}$ считается менее полным, чем $\mathbf{Q_y}$, если $\mathbf{Q_x}$ и $\mathbf{Q_y}$ появляются одинаково часто, и все события возникающие в $\mathbf{Q_x}$, также возникают в $\mathbf{Q_y}$. Все Т-Паттерны попарно проверяются на полноту, и если один паттерн признан менее полным, то он удаляется из текущего множества Т-Паттернов.

Предложенный алгоритм поиска заключается в итеративном повторении процедуры конструирования и процедуры редукции Т-Паттернов, пока текущее множество Т-Паттернов не перестанет изменяться.

Несмотря на то, что алгоритм поиска Т-Паттернов и его программная реализация в виде пакета `THEME` хорошо зарекомендовали себя в исследовательских кругах, данный метод обладает рядом недостатков. Во-первых, само определение Т-Паттерна не позволяет паттерну иметь пропуски событий. Из-за этого метод становится крайне чувствителен к шуму в исходных данных, из-за чего можно пропустить крайне информативные длинные и сложные паттерны. Во-вторых, процедура редукции может оставить в текущем множестве два разны Т-Паттерна, являющихся одним и тем же поведенческим паттерном, если критические интервалы Т-Паттернов не совпадают. В-Третьих, реализация данного алгоритма существует только в дорогостоящем программном обеспечении от `Noldus`, не позволяющем менять некоторые параметры поиска паттернов.

2 Метод поиска нечетких паттернов

Как упоминалось выше, предложенный метод основывается на алгоритме, предложенном в [2]. Основной целью являлось улучшение результатов работы алгоритма на зашумленных данных. Для этого в подразделе 1.1 описан модифицированный вид

оригинальных T-Паттернов, названный P-Паттерном(probabilistic pattern), позволяющим поведенческим закономерностям иметь пропущенные акты. Основой алгоритма будет аналогичная процедура итеративного повторения шагов конструирования и редукции паттернов. Но сами шаги конструирования и редукции, претерпели изменения, определенные тем, что было модифицировано само понятия паттерна.

2.1 Поиск «значимых» максимумов функции правдоподобия

Определенная выше функция правдоподобия P-Паттерна имеет много локальным максимумов, однако при визуально рассмотрении, обычно видно несколько ярко выраженных максимумов, в точках, которые можно считать началом вхождения паттерна.На шаге поиска новых закономерностей нам придется иметь дело с конкретными моментами времени, в которых имело место начало паттерна. Поэтому требуется определить, какие именно точки максимумов действительно соответствую началам паттернов.

Предположим, что в определенный момент времени ε имело место начала модельного P-Паттрена(без пропусков событий) длины $N_{\mathbf{P}}$. Запишем функцию правдоподобия:

$$L_{\mathbf{P}}(\varepsilon) = \prod_{i=1}^{N_{\mathbf{P}}} \frac{1}{\sqrt{2\pi}\,\sigma_i} \exp\left(-\frac{\delta_i}{2\sigma_i^2}\right), \quad \text{причем} \delta_j \sim \mathcal{N}(\,0, \sigma_j),\ (j = 1, \ldots, N_{\mathbf{P}}).$$

При таких условиях вычислим математическое ожидание $L_{\mathbf{P}}(\varepsilon)$:

$$\mathbb{E}\left[L_{\mathbf{P}}(\varepsilon)\right] = \int_{-\infty}^{+\infty} \ldots \int_{-\infty}^{+\infty} L_{\mathbf{P}}(\varepsilon) \prod_{i=1}^{N_{\mathbf{P}}} \frac{1}{\sqrt{2\pi}\,\sigma_i} \exp\left(-\frac{\delta_i}{2\sigma_i^2}\right) d\delta_1 \ldots d\delta_{N_{\mathbf{P}}} =$$

$$= \frac{1}{(2\pi)^{N_{\mathbf{P}}} \prod_{i=1}^{N_{\mathbf{P}}} \sigma_i^2} \prod_{i=1}^{N_{\mathbf{P}}} \int_{-\infty}^{+\infty} \exp\left(-\frac{\delta_i^2}{\sigma_i^2}\right) d\delta_i = \frac{1}{(2\sqrt{\pi}\,)^{N_{\mathbf{P}}}\,\sigma_1 \ldots \sigma_{N_{\mathbf{P}}}}.$$

Таким образом, нами получено характеристическое значение правдоподобия P-Паттерна длины $N_{\mathbf{P}}$ с параметрами разброса $\sigma_1, \ldots, \sigma_{N_{\mathbf{P}}}$. После чего, при поиске «значимых» максимумов отсекаются все локальные максимумы, значение которых меньше $\gamma\,\mathbb{E}\left[L_{\mathbf{P}}(\varepsilon)\right]$, где γ — заданное число от 0 до 1, в наших экспериментах обычно подходит значение $\gamma = 0.4$. Если среди оставшихся максимумов есть точки, которые удалены друг на друга меньше длины паттерна, то берется наибольший из них. Этот

шаг объясняется тем, что один P-Паттерн не может иметь место в один и тот же момент времени.

2.2 Процедура конструирования P-Паттернов

Рассмотрим два P-Паттерна $\mathbf{P}_L$(левый) и $\mathbf{P}_R$(правый). Пусть $\{\alpha_i\}_{i=1,\ldots,N_L}$ и $\{\beta_j\}_{j=1,\ldots,N_R}$ — значения в «значимых» максимумах функций правдоподобия $L_{\mathbf{P}_L}^{(end)}$ и $L_{\mathbf{P}_R}$ соответственно. Заметьте, что правдоподобие левого паттерна отсчитывается с конца, так как мы ищем связь между концом левого паттерна и началом правого. Также пусть, $\{x_{L,i}\}_{i=1,\ldots,N_L}$ и $\{x_{R,j}\}_{j=1,\ldots,N_R}$ — индексы этих «значимых» максимумов. N_L и N_R — количество таких максимумов у паттернов $\mathbf{P}_L$ и $\mathbf{P}_R$ соответственно. Определим множество межточечных расстояний:

$$\rho = \{x_{R,j} - x_{L,i} \mid x_{R,j} \geqslant x_{L,i}\}.$$

Для каждого расстояния из этого множества введем соответствующий вес:

$$w_l = \ln(1 + \alpha_i \beta_j), \quad l = 1, \ldots, M,$$

где $M = |\rho|$.

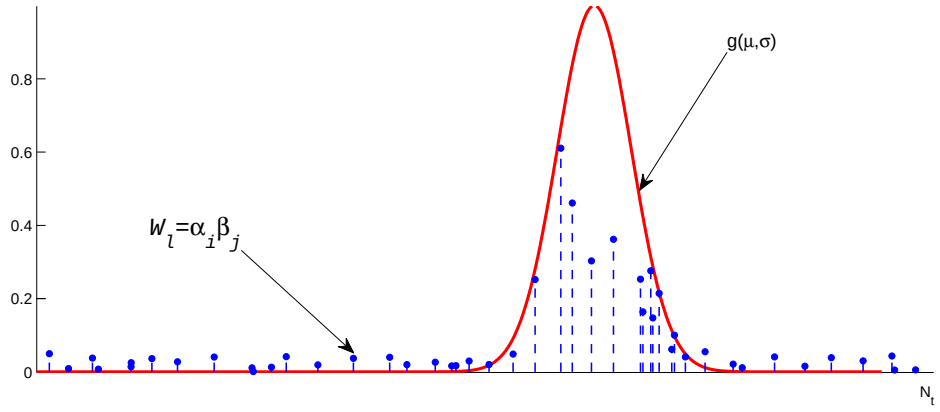


Рис. 8: Пример распределения межточечных расстояний и статистической модели связи.

Примем гипотезу, что координаты «значимых» максимумов распределены независимо и равномерно на всем наблюдаемом промежутке, т.е. $x_{L,i} \sim \mathcal{U}[0, N_t], \quad x_{R,j} \sim$

$\mathcal{U}[0, N_t]$. Тогда очевидно, что плотность распределение введенных выше межточечных расстояний имеет следующий вид:

$$p(x_R - x_L \mid x_R \geqslant x_L) = p_{LR}(x) = \begin{cases} (N_t - x)\,\frac{2}{N_t^2}, & x \in [0, N_t], \\ 0, & x \notin [0, N_t]. \end{cases}$$

Введем статистическую модель связи между паттернами(проверяемые параметры связи μ и σ фиксированы):

$$g_{\mu,\sigma}(x_i) = \frac{1}{\sqrt{2\pi}\,\sigma} \exp\left(-\frac{(x_i - \mu)^2}{2\sigma^2}\right).$$

Рассмотрим следующую сумму:

$$k = \sum_{i=1}^{M} w_i g_{\mu,\sigma}(x_i) = \sum_{i=1}^{M} \xi_i, \tag{2}$$

где $x_i \sim p_{LR}$. О распределении значений w_i мы не можем делать никаких предположений, поэтому в вычислениях будем использовать выборочные моменты.

Для оценки распределения случайной величины k проведем требуемые вычисления.

$$\mathbb{E}\left[g_{\mu,\sigma}\right] = \int_0^{N_t} g_{\mu,\sigma}(x)\, p_{LR}(x)\, dx = \int_0^{N_t} g_{\mu,\sigma}(x)\,(N_t - x)\,\frac{2}{N_t^2}\, dx \approx$$

$$\approx \frac{2}{N_t}\,\frac{1}{\sqrt{2\pi}\,\sigma} \int_{-\infty}^{+\infty} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) dx - \frac{2}{N_t^2}\,\frac{1}{\sqrt{2\pi}\,\sigma} \int_{-\infty}^{+\infty} x\, \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) dx =$$

$$= \frac{2}{N_t}\left(1 - \frac{\mu}{N_t}\right).$$

$$\mathbb{E}\left[g_{\mu,\sigma}^2\right] = \int_0^{N_t} g_{\mu,\sigma}^2(x)\, p_{LR}(x)\, dx = \int_0^{N_t} g_{\mu,\sigma}^2(x)\,(N_t - x)\,\frac{2}{N_t^2}\, dx \approx$$

$$\approx \frac{2}{N_t}\,\frac{1}{2\pi\sigma^2} \int_{-\infty}^{+\infty} \exp\left(-\frac{(x-\mu)^2}{\sigma^2}\right) dx - \frac{2}{N_t^2}\,\frac{1}{2\pi\sigma^2} \int_{-\infty}^{+\infty} x\, \exp\left(-\frac{(x-\mu)^2}{\sigma^2}\right) dx =$$

$$= \frac{1}{N_t\sqrt{\pi}\,\sigma}\left(1 - \frac{\mu}{N_t}\right).$$

$$\mathsf{var}\left[g_{\mu,\sigma}\right] = \mathbb{E}\left[g_{\mu,\sigma}^2\right] - \left(\mathbb{E}\left[g_{\mu,\sigma}\right]\right)^2 = \left(1 - \frac{\mu}{N_t}\right)\left(\frac{1}{N_t\sqrt{\pi}\,\sigma} - \frac{\mu}{{N_t}^2}\left(1 - \frac{\mu}{N_t}\right)\right).$$

Далее считаем параметры случайной величины $\xi_i = w_i g_{\mu,\sigma}(x_i)$. Здесь $\overline{w}$ и $\hat{w}$ — выборочное среднее и дисперсия весов, соответственно.

$$\mathbb{E}\left[\xi_i\right] = \mathbb{E}\left[w g_{\mu,\sigma}(x)\right] = \mathbb{E}\left[w_i\right]\mathbb{E}\left[g_{\mu,\sigma}(x_i)\right] \approx \overline{w}\,\frac{2}{N_t}\left(1 - \frac{\mu}{N_t}\right).$$

Запишем дисперсию произведения независимых случайных величин:

$$\mathsf{var}\left[\xi_i\right] = \mathsf{var}\left[w\, g_{\mu,\sigma}\right] = \left(\mathbb{E}\left[g_{\mu,\sigma}\right]\right)^2 \hat{w} + \mathbb{E}\left[g_{\mu,\sigma}\right]\overline{w}^2 + \hat{w}\,\mathsf{var}\left[g_{\mu,\sigma}\right].$$

Тогда по Центральной Предельной Теореме:

$$\sum_{i=1}^{M} \xi_i = k \;\sim\; \mathcal{N}\left(\mu_*, \sigma_*^2\right), \text{ где}$$
$$\mu_* = M\,\mathbb{E}\left[\xi_i\right],$$
$$\sigma_*^2 = M\,\mathsf{var}\left[\xi_i\right].$$

Теперь можно провести сравнение подсчитанного на реальных данных значения k с правым односторонним квантилем уровня α. Если односторонняя гипотеза о равномерном распределении событий с моделью связи $g_{\mu,\sigma}$ отвергается, то из паттернов $\mathbf{P}_L = \mathbf{e}_1^L[\,\mu_1^L, \sigma_1^L] \ldots \mathbf{e}_{N_{\mathbf{P}_L}}^L[\,\mu_{N_{\mathbf{P}_L}}^L, \sigma_{N_{\mathbf{P}_L}}^L]$ и $\mathbf{P}_R = \mathbf{e}_1^R[\,\mu_1^R, \sigma_1^R] \ldots \mathbf{e}_{N_{\mathbf{P}_R}}^R[\,\mu_{N_{\mathbf{P}_R}}^R, \sigma_{N_{\mathbf{P}_R}}^R]$ конструируется новый паттерн длины $N_L + N_R$:

$$\mathbf{P} = \mathbf{e}_1^L[\,\mu_1^L, \sigma_1^L] \ldots \mathbf{e}_{N_{\mathbf{P}_L}}^L[\,\mu_{N_{\mathbf{P}_L}}^L, \sigma_{N_{\mathbf{P}_L}}^L]\, \mathbf{e}_1^R[\,\mu, \sigma] \ldots \mathbf{e}_{N_{\mathbf{P}_R}}^R[\,\mu_{N_{\mathbf{P}_R}}^R, \sigma_{N_{\mathbf{P}_R}}^R].$$

Если существует несколько пар μ и σ, для которых отвергается гипотеза, то для конструирования P-Паттернов берутся непересекающиеся параметры(в смысле, что $[\,\mu' - 3\sigma', \mu' + 3\sigma'] \cap [\,\mu'' - 3\sigma'', \mu'' + 3\sigma''] = \varnothing$) соответствующие максимальным значениям k.

Отметим, что замена пределов интегрирования(для $\mathbb{E}\left[g_{\mu,\sigma}\right]$ и $\mathbb{E}\left[g_{\mu,\sigma}^2\right]$) с 0 до N_t на всю прямую, дает плохой результат, когда μ близко к N_t. Данный случай не критичен при поиске поведенческих закономерностей, так как на практике обычно не требуется искать такие длинные связи: даже если они будут найдены, то этот паттерн будет состоять не более, чем из двух элементов. Однако при проверке таких пар μ и σ

можно вычислять точное значение интеграла через функцию ошибок:

$$\mathbb{E}\left[g_{\mu,\sigma}\right] = \int_0^{N_t} g_{\mu,\sigma}(x)\,(N_t - x)\,\frac{2}{N_t^2}\,dx =$$
$$= \frac{\mu - N_t}{N_t^2}\left[\mathsf{erf}\left(\frac{\mu - N_t}{\sqrt{2}\,\sigma}\right) - \mathsf{erf}\left(\frac{\mu}{\sqrt{2}\,\sigma}\right)\right] +$$
$$+ \frac{1}{N_t^2}\exp\left(-\frac{\mu^2 + N_t^2}{2\sigma^2}\right)\sqrt{\frac{2}{\pi}}\,\sigma\left(\exp\left(\frac{\mu N_t}{\sigma^2}\right) - \exp\left(\frac{N_t^2}{2\sigma^2}\right)\right),$$

$$\mathbb{E}\left[g_{\mu,\sigma}^2\right] = \int_0^{N_t} g_{\mu,\sigma}^2(x)\,(N_t - x)\,\frac{2}{N_t^2}\,dx =$$
$$= \frac{1}{2\pi\sigma N_t^2}\left[\sqrt{\pi}\,(\mu - N_t)\left(\mathsf{erf}\left(\frac{\mu - N_t}{\sigma}\right) - \mathsf{erf}\left(\frac{\mu}{\sigma}\right)\right) +\right.$$
$$\left. +\sigma\left(\exp\left(-\frac{(\mu - N_t)^2}{\sigma^2}\right) - \exp\left(-\frac{\mu^2}{\sigma^2}\right)\right)\right].$$

2.3 Процедура редукции множества P-Паттернов

Так же как и в методе Магнуссона, наш алгоритм может конструировать P-Паттерны являющиеся дубликатами, или неполными копиями уже обнаруженных P-Паттернов. Далее дадим более полное описание этой проблемы.

Паттерны-дубликаты:
Один и тот же паттерн может быть сконструирован из разных подпаттернов. Например P-Паттерн $\mathbf{e}_A\mathbf{e}_B\mathbf{e}_C\mathbf{e}_D$ может быть получен как путем соединения паттернов $\mathbf{e}_A\mathbf{e}_B$ и $\mathbf{e}_C\mathbf{e}_D$, так и паттернов $\mathbf{e}_A\mathbf{e}_B\mathbf{e}_C$ и $\mathbf{e}_D$. Обычно такие P-Паттерны представляют одну и ту же поведенческую закономерность, но из-за сложного процесса соединения паттернов, функции правдоподобия этих двух паттернов-дубликатов могут отличаться.

Неполные копии паттернов:
Конструируя паттерны из подпаттернов, некоторые подпаттерны могут появляться в данных только в составе более сложных составных паттернов(см. Рис. 9. Такие подпаттерны не имеет смысла рассматривать отдельно, поэтому их надо удалять из множества найденных P-Паттернов.

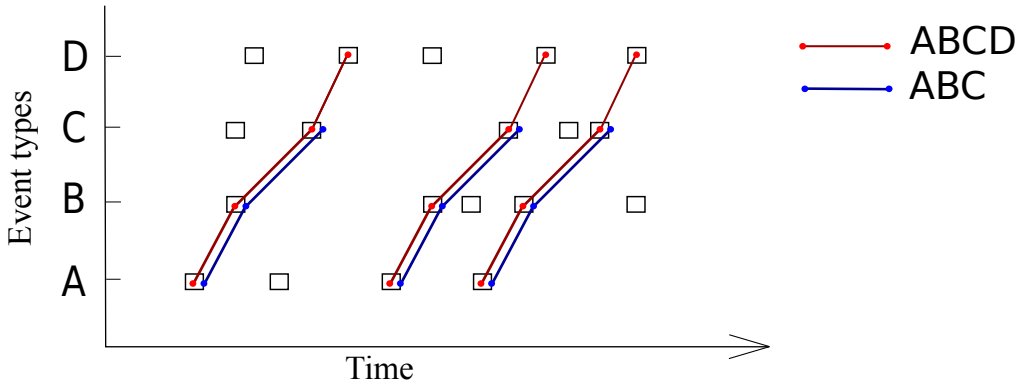


Рис. 9: Здесь BCD — неполная копия паттерна $ABCD$. Заметьте, что правдоподобие $ABCD$ отсчитанное от второго события будет похоже на правдоподобие паттерна BCD отсчитанное от первого события.

Для удаления таких P-Паттернов предлагается анализировать коэффициент корреляции функций правдоподобия. Пусть $\overrightarrow{L_{\mathbf{P},i}}$ —вектор-столбец значений функции правдоподобия, отсчитанной от i-го события во всех моментах времени наблюдения.

$$cor\left(\overrightarrow{L_1}, \overrightarrow{L_2}\right) = \frac{\overrightarrow{L_1}^T \overrightarrow{L_2}}{\sqrt{\overrightarrow{L_1}^T \overrightarrow{L_1}} \sqrt{\overrightarrow{L_2}^T \overrightarrow{L_2}}} \in [0, 1]$$

— коэффициент корреляции между двумя P-Паттернами. Чем он ближе к 1, тем два паттерна более близки друг к другу.

Процедура редукции паттернов выглядит следующим образом: перебираем все пары $\mathbf{P}_L, \mathbf{P}_R$ из множества найденных паттернов. Если все поведенческие акты, присутствующие в паттерне $\mathbf{P}_L$ также присутствуют в $\mathbf{P}_R$ с учетом порядка, и

$$\exists m\colon\ cor\left(\overrightarrow{L_{P_L,1}}, \overrightarrow{L_{P_R,m}}\right) > \nu,$$

тогда паттерн $\mathbf{P}_L$ удаляется из множества найденных паттернов. Отметим, что в данном методе не удаляются псевдопаттерны, так как они могут быть необходимы для конструирования закономерностей на следующих шагах.

2.4 Алгоритм поиска P-Паттернов

После описания основных шагов конструирования и редукции P-Паттернов, можно описать сам алгоритм поиска:

1. Инициализировать текущее множество паттернов псевдопаттернами.

Параметр	**Возможные значения**	**Значения по-умолчанию**	**На что влияет**
α	$[0, 1]$	0.001	Уровень значимости паттерна
N_{min}	$[0, +\infty]$	3	Минимальное количество появлений паттерна в данных
λ	$[0, +\infty]$	8	Допустимая степень нечеткости паттерна
ν	$[0, 1]$	0.6	Минимальная степень похожести паттернов для удаления
γ	$[0, 1]$	0.4	Чувствительность к отклонению от ожидаемого правдоподобия

Таблица 1: Параметры алгоритма поиска P-Паттернов.

2. Для всевозможных пар паттернов из текущего множества для которых не было произведено попытки их слития, провести процедуру конструирования паттернов. Сконструированные паттерны, которые встречаются в данных не менее N_{min} раз, добавить в текущее множество.

3. Для всевозможных пар паттернов из текущего множества, провести процедуру редукции паттернов.

4. Если текущее множество паттернов изменилось, перейти к п.2.

Очевидно, что описанный выше метод остановится, так как на каждом шаге будут произведены попытки сконструировать паттерны все бóльшей длины, а одни и те же паттерны не проверяются больше одного раза.

Предложенный в данной работе алгоритм обладает рядом параметров, которые не настраиваются автоматически. Эти параметры должны быть выставлены вручную исследователями, исходя из априорных сведений о типе исходных данных, или ожидаемого вида поведенческих закономерностей. Список этих структурных параметров с объяснениями и значениями по-умолчания представлен в Таблицу 1.

Отметим, что предложенный алгоритм является отчасти переборным, а самыми вычислительно-сложными процедурами являются подсчет функции правдоподобия для каждого паттерна, отсчитанная для каждого события, и подбор параметров смещения и разброса при попытке конструирования паттернов. Для ускорения работы, была реализованная параллельная версия данных процедур, описанная в подразделе **??**.

2.5 Случайные паттерны

Когда исследования проводятся на больших наборах данных, то закономерности могут возникать даже, если выборка была сгенерированна случайно. Поэтому, для найденного множества паттернов было бы полезно оценить, являются ли они случайными, или «структурными». Одним из подходов к решению данной задачи, является анализ рандомизированных данных:
для каждого массива данных создаются данные с такой же протяженностью периода наблюдения и мощностью множества допустимых событий, но для каждого допустимого события $\mathbf{e}_i$, моменты времени его появления генерируются случайно из равномерного на $[0, N_t]$ распределения с вероятностью $N_{\mathbf{e}_i}/N_t$. Для полученных данных применяется процесс поиска паттернов с *теми же параметрами*, которые применялись на исходных данных. Описанные действия исполняются несколько раз, после чего результаты поиска на рандомизированных данных сравниваются с исходными.

Считается, что поиск паттернов прошел успешно, если в исходных данных было выявлено значительно больше закономерностей, чем в рандомизированных, или они оказались длиннее.

3 Параллельная реализация методов поиска закономерностей

Как было отмечено выше, рассмотренные алгоритмы поиска закономерностей имеют переборную составляющую. Так как при более полном переборе вариантов, позволяющем находить более длинные и значимые паттерны, время работы алгоритма увеличивается, то имеет смысл произвести распараллеливание метода. В общем

случае, при полном переборе всевозможных T-Паттернов, алгоритм имеет экспоненциальную сложность от количества возникших событий в данных(см. Рис. 10). Возможны два подхода к параллельной реализации данного алгоритма.

- Во-первых, можно разбивать каждую итерацию алгоритма на, очевидно, независимые подзадачи сравнения пар паттернов(на шагах конструировании, редукции паттернов, и вычислении правдоподобия каждого паттерна). В этом случае, подзадачи вообще не требуют пересылки данных между собой, кроме как для инициализации и возвращения результата работы. Получается, что мы, фактически, одновременно обрабатываем несколько паттернов на разных вычислительных узлах. Такой подход можно реализовать на современных многоядерных системах с общей памятью(symmetric multiprocessing), или на вычислительных кластерах.

- Во-вторых, можно распараллеливать саму процедуру конструирования и вычисления правдоподобия паттернов. Тогда все паттерны будут обрабатывать последовательно, друг за другом, но очень быстро из-за того, что проверка реализована параллельно. Такой подход не может быть эффективно реализован, на вычислительном кластере, из-за большого количества пересылок данных между вычислительными узлами. Возможно более эффективная реализация для многоядерной архитектуры, но так как исходная задача разбивается на крайне простые подзадачи, то самым уместным решением будет использование современных программируемых графических чипов, работающих по схеме SIMD(Single Instruction Multiple Data).

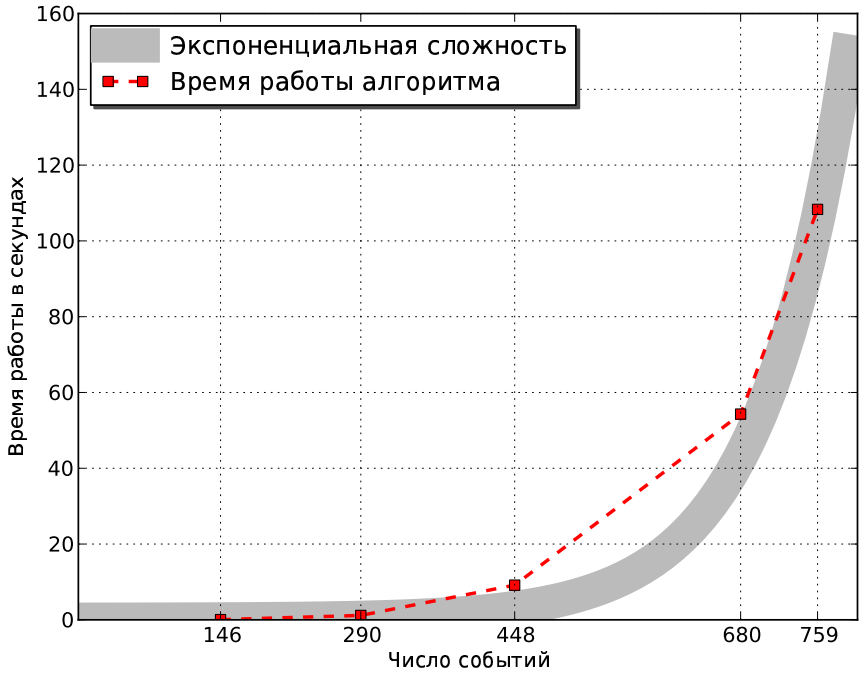


Рис. 10: Сложность алгоритма поиска T-Паттернов в зависимости от количества событий в данных.

3.1 Параллельная реализация алгоритма поиска «четких» закономерностей(T-Паттерны)

Для параллельной реализации метода поиска T-Паттернов, предложенного Магнуссоном, мы первый подход, где алгоритм разбивается на подзадачи обработки паттернов. Последовательная версия алгоритма была реализована нами ранее на языке `C`, поэтому мы произвели распараллеливание метода для архитектуры SMP с помощью `OpenMP`.

Результаты полученного ускорения представлены на графике 11 и в Таблице 2 Техническая реализация была достаточна тривиальна: этапы конструирования и редукции T-Паттернов были помещены в параллельную секцию с описанием локальных и общих участков памяти, после чего на каждом шаге проводится сбор результатов работы со всех вычислительных узлов и модификация текущего множества найденных T-Паттернов. Распределение подзадач по вычислительным узлам производит встроенный в `OpenMP` планировщик, простой вычислительных узлов возможен,

только в конце каждой итерации алгоритма, когда не остается необработанных пар паттернов.

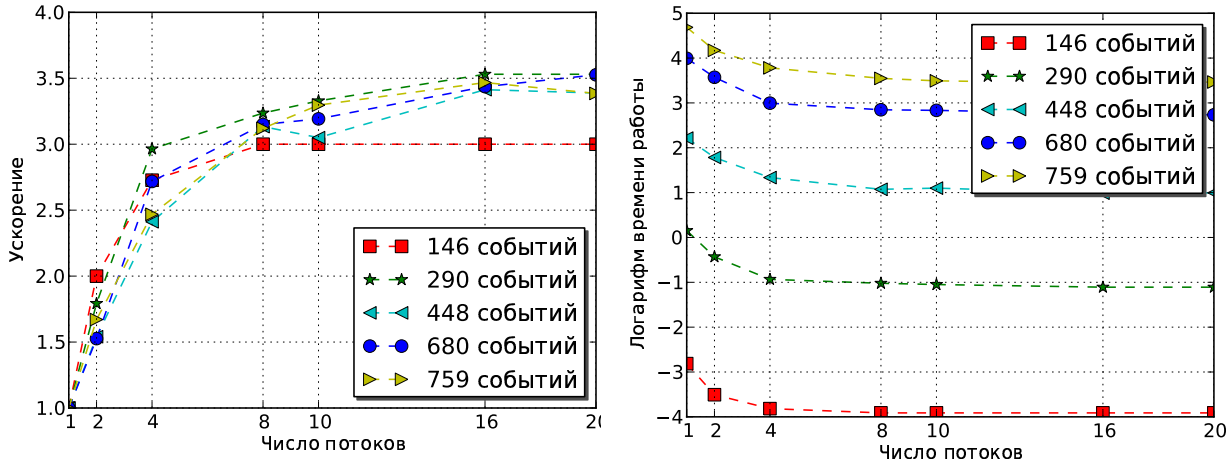


Рис. 11: Время работы и ускорение алгоритма поиска T-Паттернов на 4-х ядерном процессоре.

На большинстве примеров видно, что увеличение количества потоков ожидаемо уменьшает время работы программы. Как и стоило ожидать, на задачах с большим количеством событий, мы получаем большее ускорение при большом количестве узлов. Так как эксперименты производились на 4-х ядерном процессоре, то при количестве потоков большем, чем 4 ускорение перестает быть линейным.

3.2 Параллельная реализация алгоритма поиска «нечетких» закономерностей(P-Паттерны)

Алгоритм поиска P-Паттернов является еще более вычислительно-сложным, чем алгоритм поиска T-Паттернов, поэтому для его применения на реальных данных крайне важна параллельная реализация. Для распараллеливания алгоритма была использована технология `NVIDIA CUDA`, позволяющая создавать программы, в том числе, для современных игровых видеокарт, которые широко распространены и имеют один из самых выгодных показателей FLOPS(FLoating point OPerations per Second) за доллар.

Подробное описание архитектуры `GPU` от `NVIDIA` и технологии `CUDA` можно найти, например, в [11, 13]. Отметим лишь, что логически все вычисления на `GPU` разбива-

ются на блоки(*blocks*), блоки, в свою очередь, разбиваются на нити(потоки, *threads*). Нити из одного блока могут синхронизироваться и обмениваться данными через высокоскоростную shared память. Нити из одного блока физически выполняются на одном мультипроцессоре *half-warp*'ами(группы по 16 нитей), что нужно учитывать при доступе к различным видам памяти. Также существуют доступные и крайне эффективные [13, 12] схемы одного из важнейших параллельных примитивов: параллельной редукции. Крайне важно избегать ветвлений внутри *half-warp*'ов, так как мультипроцессор исполняет одновременно одни и те же операции для нескольких разных данных(схема SIMD).

При этом, между нитями на `CPU` и нитями на `GPU` есть принципиальные различия [13]:

- нити на `GPU` обладают крайне «небольшой стоимостью» — их создание и управление требует минимальных ресурсов (в отличии от `CPU`);
- для эффективной утилизации возможностей `GPU` нужно использовать многие тысячи отдельных нитей (для `CPU` обычно нужно не более 10-20 нитей)

В нашей реализации на `GPU` были перенесены две самые вычислительно-сложные процедуры: конструирование паттернов, вычисление правдоподобия паттерна. Предложенная конкретная реализация проектировалась с учетом спецификаций устройств `NVIDIA Compute Capability 1.1`. Данная спецификация определяет, например, доступные инструкции чипа, объем типов памяти, количество регистров.

Для параллельной версии конструирования паттернов использовался следующий подход: разные блоки соответствуют разным *парам* тестируемых параметров μ и σ для двух паттернов. В shared память каждого блока загружается множество межточечных расстояний, после чего эффективным образом вычисляется статистика k (2) для каждой данной пары параметров. Данная статистика сравнивается с фиксированным порогом и принимается решение о добавлении паттерна в текущее множество.

Для параллельной версии процедуры вычисления правдоподобия использовался следующий подход: моменты времени, в которых вычисляется правдоподобие разбиваются на участки, соответствующие блокам, каждый участок, в свою очередь,

состоит из моментов времени, каждый из которых соответствует нити внутри блока. Далее, производится процесс редукции для поиска ближайшего элемента паттерна и по формуле (1).

Вычислительные эксперименты(см. Рис. 12,13) проводились на устройстве `NVIDIA GeForce 8800GTX` с 128 потоковыми процессорами, вычисления проводились с одинарной точностью(single precision floating-point format). Для конструирования паттернов удалось достичь ускорения работы в 15–20 раз, а для процедуры вычисления правдоподобия было получено ускорение в 120–140 раз, что соответствует максимальному порядку ускорения задач для данной архитектуры [14]. Скорее всего, для современных `GPU`(`GeForce 8800GTX` был выпущен в 2006-ом году), имеющих большее количество потоковых процессоров и shared памяти, возможно создать еще более эффективную реализацию, которая будет проводить вычисления уже с двойной точностью(double precision floating-point format).

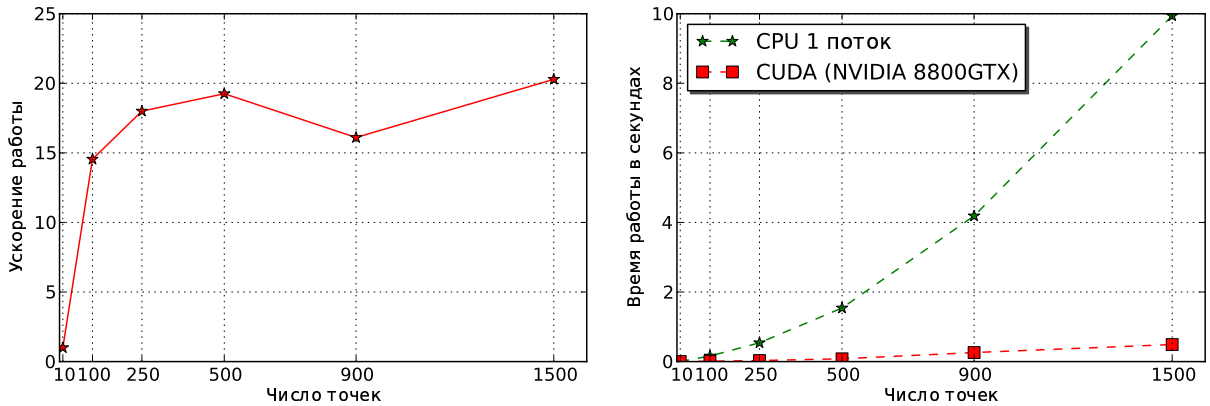


Рис. 12: Время работы и ускорение процедуры конструирования P-Паттернов на GPU и CPU.

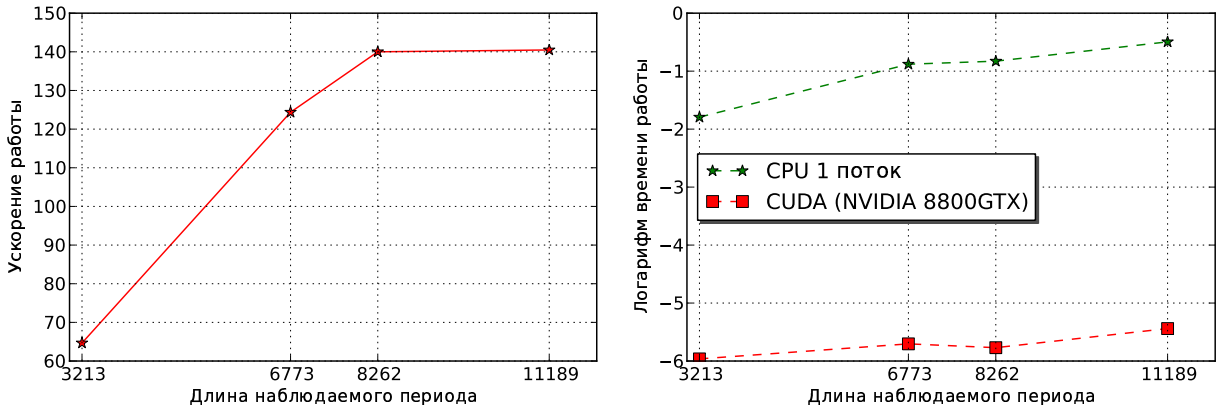


Рис. 13: Время работы и ускорение вычисления правдоподобия P-Паттернов на GPU и CPU.

4 Сравнение алгоритмов поиска T-Паттернов и P-Паттернов. Результаты работы на экспериментальных данных

4.1 Сравнение результатов алгоритмов поиска T-Паттернов и P-Паттернов

Предложенный в данной работе алгоритм поиска P-Паттернов создавался, как расширение исходного алгоритма поиска T-Паттернов, поэтому логично требовать, чтобы паттерны найденные исходным алгоритмом(по крайней самые длинные) также были найдены предложенным методом. Сложность заключается в том, проблематично точно сопоставить найденный T-Паттерн с найденным P-Паттерном. Данная работа была сделана вручную. Результаты показывают, что среди P-Паттернов найденных предложенным методом, в среднем, присутствует 93.6% T-Паттернов, найденных исходным методом(см. Рис. 14). Более того, предложенный метод находит более длинные закономерности(см. Рис. 14). На модельных зашумленных данных, как и ожидалось, метод поиска P-Паттернов находит более длинные версии T-Паттернов(пример на Рис. 15).

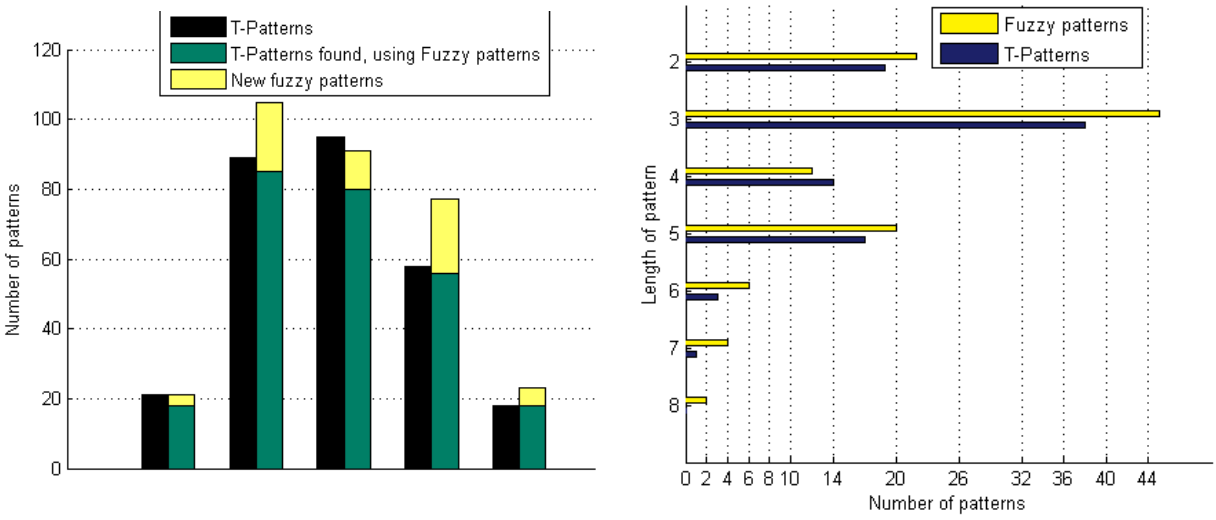


Рис. 14: Слева: количество паттернов, найденных разными методами. Справа: распределение длин найденных паттернов.

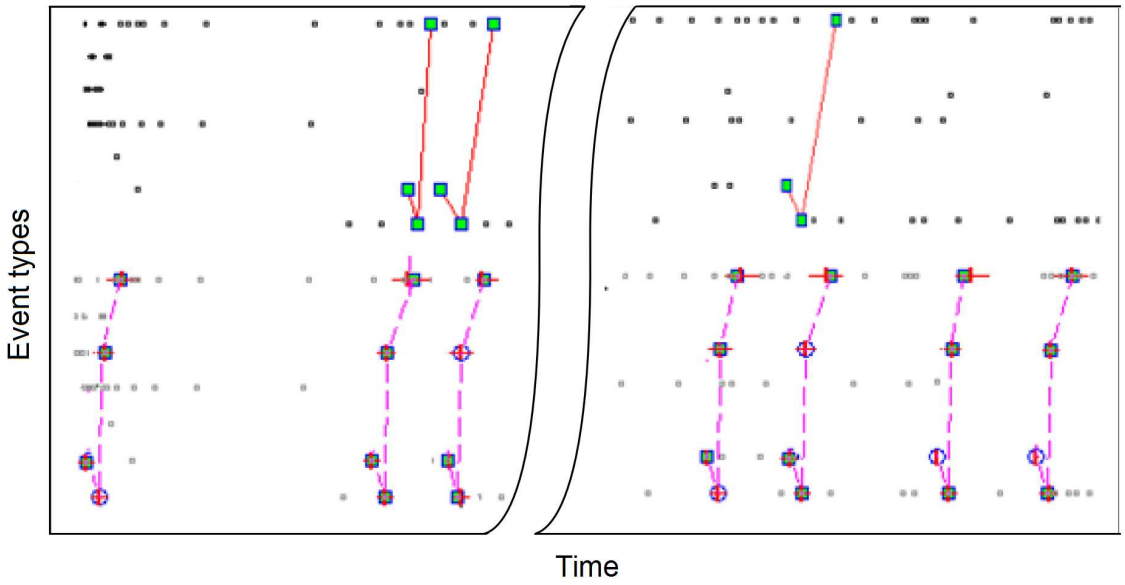


Рис. 15: Сравнение найденных закономерностей в реальных данных, используя алгоритм поиска Т-Паттернов(сверху) и алгоритм поиска Р-Паттернов(снизу). Пропущенные события показаны полыми кружками. Р-Паттерны имеют бóльшую длину и чаще встречабтся в данных. Этот объясняется тем, что в данных не присутствует достаточно экземпляров паттерна без пропусков.

4.2 Эксперименты с мышами

TODO

4.3 Обсуждения и выводы

Реализованный в данной дипломной работе метод решает поставленные перед ним задачи и производит качественный поиск закономерностей как в синтетических временны́х рядах, так и в реальных поведенческих данных. Предложенный метод поиска P-Паттернов имеет 2 главных недостатка:

– из-за статистической основы метода, при малых объемах данных, некоторые паттерны могут быть приняты за шум. С этим эффектом можно бороться путем более тщательного подбора параметров алгоритма.

– Алгоритм имеет экспоненциальную сложность и, несмотря на 140-кратное ускорение, будет крайне долго работать на очень длинных данных(на несколько порядков длиннее, чем стандартная разметка поведения мышей). Имеющиеся в нашем распоряжении поведенческие данные обрабатывались не дольше двух минут.Если данная проблема будет актуальна, ее можно решить, выставив ограничение на максимальную длину связи между событиями в паттерне.

В данной работе алгоритм поиска закономерностей рассматривался в контексте применения его для анализа поведения. Но, очевидно, заменив понятие поведенческого акта на какое-то абстрактное событие(маркер), мы можем искать закономерности в различных потоках данных. Например, событиями могут быть: повышения и понижения курсов валют в анализе поведения рынка; аминокислоты, кодоны, или азотистые основания при анализе структуры ДНК; всплеск активности отдельного нейрона при анализе спайковой активности нейронных культур; новостные тренды(в виде ключевых слов) при анализе закономерностей в политике и обществе. Например, алгоритм T-Паттернов и ранее был использован для анализа структуры ДНК [15] и стратегии футбольных команд во время матчев [16]

5 Заключение

Основные результаты работы:

- Разработан метод для поиска закономерностей в последовательностях событий.

- Для поиска поведенческих закономерностей предложенный метод дает результаты, как минимум, не хуже, чем широко зарекомендовавшие себя методы.

- Представлены свободные, документированные реализации методов поиска предложенного метода поиска P-Паттернов, а также метода поиска T-Паттернов.

- Разработаны эффективные параллельные версии методов поиска P-Паттернов и T-Паттернов.

    - Пиковое ускорение для параллельной реализации поиска T-Паттернов на 4-х ядерном `CPU` — 3.53 раза.

    - Пиковое ускорение для параллельной реализации процедур поиска P-Паттернов на `NVIDIA GeForce8800GTX` — 20 раз(этап конструирования) и 140 раз(этап подсчета правдоподобия).

- Для реальных экспериментов с мышами, предложенный метод выделяет паттерны, по которым можно определить из какой группы была взята наблюдаемая особь. То есть, решена задача классификации особей по группам, зная только их поведение.

Система, разработанная и реализованная в данной работе, позволит биологам решать ряд практически важных задач, возникающих при исследовании поведения: выделение поведенческих закономерностей животных, анализ вариабельности и сложности поведения, выявление отличительных черт поведения среди разных групп животных, анализ спайковой активности нейронов. Решение этих задач позволит проводить качественный анализ влияния медицинских препаратов на поведение животных, связи различных анатомических структур мозга и поведения.

Дальнейшая работа в данной области будет связана, во-первых, с исследованием возможности расширения множества задач, которые можно решить предложенным методом. Во-вторых, планируется разработать удобную для биологов графическую среду для исследования поведенческих закономерностей.

Авторы статьи выражают благодарность членам «Лаборатории Нейробиологии Памяти» при Институте Нормальной Физиологии П.К. Анохина, возглавляемой

членом-корреспондентом РАН К.В. Анохиным. Отдельно благодарим Ирину Зарайскую, предоставившую нам экспериментальные данные по поведению мышей без гиппокампа.

Работа выполнена при финансовой поддержке РФФИ(проект №08-01-00405), гранта Президента РФ(МК3827.2010.9), и федеральной целевой программы «Научные и научно-педагогические кадры инновационной России» на 2009–2013 годы(контракт №П1265).

Список литературы

[1] P. Martin, P. Bateson. Measuring Behaviour: An Introductory Guide. — Cambridge University Press, second edition, 1993.

[2] M.S. Magnusson. Discovering hidden time patterns in behavior: T-patterns and their detection. — Behavior Research Methods, Instruments, Computers 2000, 32 (I), 93–I IO.

[3] V.V. Vishnevskiy, D.P Vetrov. The Algorithm for Detection of Fuzzy Behavioral Patterns. — Proceedings of Measuring Behavior 2010,ISBN 978-90-74821-86-5.

[4] R. Stoop, B. Arthur. Periodic orbit analysis demonstrates genetic constraints, variability, and switching in Drosophila courtship behavior. // Chaos – 2008 – vol.18/2.

[5] P. Cvitanovic. Periodic orbits as the skeleton of classical and quantum chaos. // Physica D: Nonlinear Phenomena – 1991 – vol.51 – Issues 1-3 – Pp. 138–151.

[6] А.М. Шурыгин. Математические методы прогнозирования. — Горячая Линия – Телеком, 2009. ISBN 978-5-9912-0062-2.

[7] N. Raichma, E. Ben-Jacob. Identifying repeating motifs in the activation of synchronized bursts in cultured neuronal networks. // Journal of Neuroscience Methods – 2007 – vol.170 – Pp. 96—110.

[8] J.-M. Fellous, P.H.E. Tiesinga, P.J. Thomas, T.J. Sejnowski. Discovering spike patterns in neuronal responses. // The Journal of Neuroscience – March 24, 2004 – vol.24(12) – Pp. 2989-–3001.

[9] Sheng Ma, J.L. Hellerstein. Mining partially periodic event patterns with Uuknown periods. — IBM T.J. Watson Research Center Hawthorne, NY 10532

[10] В.В. Вишневский, Д.П. Ветров. Поиск скрытых поведенческих паттернов. — Курсовая работа. ВМиК МГУ 2009.

[11] NVIDIA Corporation. NVIDIA CUDA. Compute Unified Device Architecture. — [PDF] – 2008 – (NVIDIA_CUDA_Programming_Guide_2.0.pdf).

[12] M. Harris. Optimizing parallel reduction in CUDA. — [PDF] – (reduction.pdf).

[13] Б.А. Боресков. Основы CUDA. — [HTML] – (http://steps3d.narod.ru/tutorials/cuda-tutorial.html, http://steps3d.narod.ru/tutorials/cuda-2-tutorial.html).

[14] CUDA Zone. CUDA community showcase. — [HTML/SWF] – (http://www.nvidia.com/object/cuda_apps_flash_new.html).

[15] M.S. Magnusson. Analyzing complex real-time streams of behavior: repeated patterns in behavior and DNA. — L'éthologie appliquée aujourd'hui. – vol.3 – Ethologie humaine. Levallois-Perret – France, 2003 – ISBN 2-7237-0025-9.

[16] G.K. Jonsson, S.H. Bjarkadottir, B. Gislason. Detection of real-time patterns in sports: interactions in football. — L'éthologie appliquée aujourd'hui. – vol.3 – Ethologie humaine. Levallois-Perret – France, 2003 – ISBN 2-7237-0025-9.

6 Приложение 1. Документация к программной реализации алгоритма алгоритмов поиска P-Паттернов и T-Паттернов. Интерфейс в среде MATLAB

Все исходные коды, бинарные пакеты, и данные могут быть запрошены по адресу valera.vishnevskiy@yandex.ru.

6.1 Формат входных данных

Временной ряд должен быть представлен в виде текстового файла следующего формата:

```
Time    Event
0       :
time(int)    event(string_w\o_spaces)
...
time(int)    event(string_w\o_spaces)
time(int)    &
```

Обратите внимание на то, что элементы каждой строки разделены символом табуляции \t.

6.2 Консольное приложение для поиска T-Паттернов

Консольное приложение написано на чистом C с расширением OpenMP, для создания исполняемого файла использовался компилятор GCC 4.4.5(arch x86_64-linux-gnu). Приложение можно запускать из консоли следующим образом: #>./t_pattern_core -i *путь_к_входному_файлу* -o *путь_к_выходному_файлу* [-s *уровень_значимости_паттернов*] [-n *минмиальное_число_появления_паттерна*] [-q ci_shortest|ci_longest|ci_most_significant] [-e]

Переменная окружения OMP_NUM_THREADS определяет количество потоков на которое распараллеливается программаю

6.3 Интерфейс к приложению для поиска T-Паттернов в среде MATLAB

`function [events, Nt, ts] = T_GENERATE_PATTERN( pat_sym, noise_sym, CIs, Npat, dist_b_patterns, Pnoise1, Pnoise2 )`

Параметр	**Описание**
Вход:	
`pat_sym`	Матрица `1x`N_p типа `char`, определяющая паттерн. Каждый символ — событие.
`noise_sym`	Матрица `1xN` типа `char`. Определяет события которые будут генерироваться случайно.
`CIs`	Матрица (N_p`-1)x2` типа `int`. В i-й строке которой, записан соответствующий критический интервал.
`Npat`	Количество паттернов, которые требуется сгенерировать.
`dist_b_patterns`	Максимальное расстояние между двумя появлениями паттерна.
`P_noise1`	Частота встречаемости шумовых символов.
`P_noise2`	Вероятность того, что символ из паттерна будет зашумлен.
Выход:	
`events`	Массив структур `1xN`, где `N` — количество событий. Каждая структура состоит из двух полей: `event_name` — строка названия события, и `indexes` — матрица `1xN` типа `int`. Определяет времеа появления событий.
`Nt`	Продолжительность получившегося периода наблюдений.
`ts`	Символьная матрица `1xNt`.

Создает временной ряд, содержащий `Npat` копий одного искусственного паттерна.

```
function [events, Nt] = T_LOAD_FILE( fname )
```

Параметр	Описание
Вход:	
`fname`	Путь к текстовому фалу правильного формата.
Выход:	
`events`	Массив структур `1xN`, где `N` — количество событий. Каждая структура состоит из двух полей: `event_name` — строка названия события, и `indexes` — матрица `1xN` типа `int`. Определяет времеа появления событий.
`Nt`	Продолжительность получившегося периода наблюдений.

Загружает временной ряд из текстового файла, определенного выше формата.

```
function patterns = mexPattern( events, Nt, levels, allow_same_events,
ci_strategy )
```

Параметр	Описание
Вход:	
`events, Nt`	См. определение `T_GENERATE_PATTERN`.
`levels`	Vатрица `Nx3` уровней значимости. Каждая строка содержит: длину паттернов, к которым должны применяться следующие параметры; минимальный уровень значимости α; минимальное количество вхождений паттерна N_{min}.
`allow_same_events`	1 если, разрешается появление одинаковых событий в паттерне, 0 иначе.
`ci_strategy`	Стратегия выбора критического интервала. 1 — стратегия выбора длиннейшего интервала, 2 — кратчайшего, 3 — самого значимого.
Выход:	

`patterns`	Массив структур. Каждая структура описывает найденный паттерн. Поля структуры: `Events` — индексы событий, которые составляют паттерн. `CIs` — интервалы, соответствующие критическим связям между событиями в паттерне. `Sign` — уровень значимости найденного паттерна. `Nab` — количество появлений паттерна. `DS` — двойные серии(*double series*) паттерна. `String` — строка, описывающая паттерн в следующем формате: $Event_1[dL_1, dR_1]Event_2 \ldots Event_m <$ уровень значимости $> \{N_{ab}\} : (DS_1)(DS_2) \ldots (DS_{N_{ab}})$

Реализует поиск паттернов во временных рядах.

`function T_DRAW_PATTERNS( patterns, events, Nt, np )`

Параметр	**Описание**
Вход:	
`patterns, events, Nt`	См. предыдущие определения.
`np`	Номер паттерна, который нужно представить. Или -1 для последовательного вывода всех паттернов.

Строит диаграмму найденных паттернов.

`function [p] = T_STAT_VALIDATE( Nt, events, levels, nvalidations )`

Параметр	**Описание**
Вход:	
`Nt, events, levels`	См. предыдущие определения.
`nvalidations`	Количество повторений процедуры рандомизации.
Выход:	
`p`	Целочисленная матрица `1 x nvalidations`. В каждой ячейке — количество паттернов, найденных в рандомизированных данных.

Процедура статистической валидации.

6.4 Интерфейс к приложению для поиска P-Паттернов в среде MATLAB

Для каждого отдельног офайла определяется следующая структура параметров алгоритма. Параметры, помеченные как технические можно всегда оставлять заданными по-умолчанию.

`conf.alpha` *Уровень значимости паттерна* α. `[=0.001]`

`conf.kkmax` *Выравнивание весов паттернов. Техничиский.* `[=2.1]`

`conf.Nmin` *Минимальное число появления паттерна* N_{min}. `[=3]`

`conf.lambda` *Максимальный уровень нечеткости паттерна* λ. `[=8]`

`conf.lhmult` *Толерантность к отклонению от правдоподобия* γ. `[=0.5]`

`conf.cor` *Минимальный коэфф. корреляции* ν. `[=0.6]`

`conf.maxSigma` *Максимальный тестируемый разброс.* `[=round( Nt / 250 )]`

`conf.maxMu` *Максимальное тестируемое отклонение.* `[=round( Nt / 3 )]`

`conf.min_pow_missed` *Степень экспоненты при пропуске. Технический.* `[=-8]`

`function [p] = T_GENERATE_FUZZY_PATTERN( pat_sym, noise_sym, thetas, Npat, dist_b_patterns, Pnoise1, Pnoise2 )`

Параметр	**Описание**
Вход:	
`pat_sym, noise_sym, Npat, dist_b_patterns, Pnoise1, Pnoise2`	См. предыдущие определения для Т-Паттернов.
`thetas`	Матрица `Npat x 2` пар смещение, разброс для каждого события в паттерне.
Выход:	
`events`	Массив структур `1xN`, где `N` — количество событий. Каждая структура состоит из двух полей: `event_name` — строка названия события, и `indexes` — матрица `1xN` типа `int`. Определяет времеа появления событий.
`Nt`	Продолжительность получившегося периода наблюдений.
`ts`	Символьная матрица `1xNt`.

Процедура генерация данных с нечетким паттерном.

`function [ps] = T_PS_FROM_TS( events, Nt, sigma0, conf )`

Параметр	**Описание**
Вход:	
`events, Nt`	См. предыдущие определения.
`sigma0`	Значение разброса для псевдопаттерна. Обычно берется =1.
`conf`	Структура параметров, см. выше.
Выход:	
`ps`	Массив структур нечетких псевдопаттернов.

Создание множества псевдопаттернов из временного ряда.

```
function [ppl] = T_FIND_PATTERNS( events, ps, Nt, pattern_window,
use_cuda, conf )
```

Параметр	**Описание**
Вход:	
`events, Nt, ps,`	См. предыдущие определения.
`conf`	
`pattern_window`	Технический параметр. Значение не важно.
`use_cuda`	Флаг использования `GPU. true|false`.
Выход:	
`ppl`	Массив структур нечетких паттернов.

Производит поиск закономерностей.

7 Приложение 2. Таблицы

Число потоков	Число событий	Время работы	Ускорение
1	146	0.060	1.00
2	146	0.030	2.00
4	146	0.022	2.73
8	146	0.020	3.00
10	146	0.020	3.00
16	146	0.020	3.00
20	146	0.020	3.00
1	290	1.165	1.00
2	290	0.650	1.79
4	290	0.393	2.96
8	290	0.360	3.24
10	290	0.350	3.33
16	290	0.330	3.53
20	290	0.330	3.53
1	448	9.165	1.00
2	448	5.950	1.54
4	448	3.800	2.41
8	448	2.925	3.13
10	448	3.005	3.05
16	448	2.685	3.41
20	448	2.705	3.39
1	680	54.300	1.00
2	680	35.580	1.53
4	680	19.980	2.72
8	680	17.240	3.15
10	680	17.010	3.19
16	680	15.800	3.44
20	680	15.400	3.53
1	759	108.320	1.00
2	759	64.845	1.67
4	759	43.880	2.47
8	759	34.710	3.12
10	759	32.860	3.30
16	759	31.240	3.47
20	759	32.000	3.38

Таблица 2: Время работы и ускорение алгоритма поиска T-Паттернов на 4-х ядерном процессоре.

Число точек	Время работы на CPU	Время работы на GPU	Ускорение
10	0.00	0.00	1.00
100	0.16	0.01	14.55
250	0.54	0.03	18.00
500	1.54	0.08	19.25
900	4.19	0.26	16.10
1500	9.94	0.49	20.29

Таблица 3: Время работы и ускорение процедуры конструирования P-Паттернов на GPU и CPU.

Длина времени наблюдения	Время работы на CPU	Время работы на GPU	Ускорение
3213	0.166	0.002571	64.67
6773	0.415	0.003333	124.37
8262	0.437	0.003121	140.00
11189	0.610	0.004340	140.48

Таблица 4: Время работы и ускорение вычисления правдоподобия P-Паттернов на GPU и CPU.