Выявление аномалий в условиях функционирующих процессов

И. Ю. Терёхина

***Аннотация*—В данной работе рассматриваются алгоритмы для задачи поиска аномалий в случае наличия нескольких эталонных процессов при некоторых ограничениях на их структуру. В качестве формальной модели процессов рассматривается ациклический ориентированный граф.**

Рассматривается случай, когда при построении модели процесса в трассе лога может содержаться след только одного процесса. Для этого случая изучается вопрос сложности выявления аномалий, когда процессы состоят из различных действий и когда пересечение множеств действий процессов является конечным множеством.

Также рассматривается случай, когда в одной трассе содержатся следы множества процессов. Предлагается достаточное условие, накладываемое на трассы лога событий, такое, что наличие в трассах следов нескольких процессов не представляет собой усложнение рассматриваемых задач, а позволяет применить алгоритмы построения множества процессов и поиска аномалий аналогично случаю, когда в одной трассе лога может содержаться след только одного процесса.

Определены оценки сложности построения формальных моделей для процессов, в зависимости от свойств лога, также даны оценки сложности ответа на вопрос, является ли поступающая трасса аномальной относительно уже построенных моделей процессов.

***Ключевые слова*—поиск аномалий, ациклический ориентированный граф, обнаружение процесса, построение модели процесса.**

I. Введение

Задача построения модели процесса не является новой, но остается актуальной. По запросу “process model mining” с начала 2020 года в google scholar может быть найдено более 100 тысяч публикаций, например [1]–[5]. В контексте задач информационной безопасности задача построения модели процесса часто рассматривается вместе с задачей поиска аномалий.

Для задачи построения модели процесса обычно предполагается, что известен некоторый журнал событий (лог) эталонного процесса, по которому строится модель.

В качестве формальной модели могут рассматриваться: конечный автомат [6], [7], сеть Петри [8], [9], вероятностные модели [10]-[12], ориентированный граф [13]-[15], множество ассоциативных правил [16], [17] и другие. У каждой из моделей есть свои достоинства и недостатки. Так, например, модели в виде сетей Петри обладают большой выразительной мощностью, но было показано [18], что построение корректной сети Петри для лога может быть неоднозначной задачей. В задаче поиска аномалий обычно происходит сопоставление новых событий с имеющейся моделью процесса, если событие не может быть представлено в построенной модели, событие называется “аномальным”.

В данной работе рассматривается подход представления процессов в множестве ориентированных ациклических графов (DAG). Достоинством данного подхода является хорошая интерпретируемость полученной модели, линейная сложность построения модели относительно размера лога. Недостатком данного метода является ограничение, накладываемое на модель процесса, заключающееся в отсутствии ориентированных циклов, что сужает класс логов, на которых данный метод может быть применен. В настоящей работе рассматривается расширение трех алгоритмов, предложенных в работе [15], с моделирования одного до моделирования нескольких процессов в рамках одного лога событий и оценивается их сложность.

II. Определения и постановка задачи

Модель одного процесса строится в виде ориентированного графа без циклов. Вершины графа задают действия процесса, действия предполагаются атомарными. Дуги графа задают причинно-следственные связи между действиями. Таким образом, если полагается, что действие A должно быть выполнено до действия B, то в модели должна появиться дуга, ведущая из A в B. Среди множества всех ориентированных ациклических графов на модель процесса при заданном логе накладываются следующие условия:

- *Полнота* – граф должен содержать все зависимости, имеющиеся в логе;
- *“Неизбыточность”* – граф не должен порождать зависимостей, которых не было в логе процесса;
- *Минимальность* – граф должен состоять из минимального количества дуг.

В отличие от работы [15], рассматривается определение процесса без функции переходов и функции выходов. Полагается, что *процесс* P задается с помощью конечного множества действий V и ориентированного графа $G = (V, E)$.

Пусть граф G имеет один *исток* – вершину, в которую не следует ни одна дуга, есть только исходящие дуги, и

Статья получена 3 ноября 2021.

И. Ю. Терёхина, Факультет вычислительной математики и кибернетики Московского государственного университета им. М. В. Ломоносова, (email: iteryokhina@cs.msu.ru).

сток – вершину, из которой не исходит дуг. Таким образом, накладывается ограничение, что в рамках одного процесса все его исполнения начинаются с одного и того же действия и заканчиваются одним и тем же действием. Если в рамках процесса нет такого начального и конечного действий, то их можно добавить искусственно. В данной работе полагается, что каждая информационная технология имеет свое начальное и конечные действия, которые при этом не обязательно должны быть уникальными среди всех функционирующих одновременно информационных технологий.

A. Основные определения

Пусть задан конечный алфавит V возможных действий для процесса P:

Определение 1 (Лог, трасса, действие). *Трасса* – упорядоченное конечное множество, состоящее из записей вида (P, A, T), где P – идентификатор процесса, $A \in V$ – *действие*, T – время исполнения записи. *Лог* – неупорядоченное конечное множество трасс.

Далее трасса рассматривается как упорядоченное по времени конечное множество действий из алфавита V. Таким образом, полагается, что:

- Записи одной трассы могут принадлежать только одному процессу P_i, $i = 1,\ldots,s$;
- Если параметр времени T влияет на выполнение некоторого действия A, то в трассе лога будут рассматриваться действия $A_1,\ldots,A_q \in V$, где q – количество различных исходов действия A.

Например, запись лога $L = \{ABC, DE\}$ означает, что в этом логе содержатся 2 трассы и множество действий $\{A, B, C, D, E\}$.

Если в процессе существует зависимость между двумя действиями, то эти действия должны появляться в трассах лога в соответствующем порядке. Однако, о зависимостях действий в процессе возможно узнать только по тем зависимостям, которые представлены в логе процесса, поэтому определение зависимости вводится для лога событий. В модели процесса каждая зависимость представляется либо как направленная дуга, либо как направленный путь от одного действия к другому.

Определение 2 (Частичный порядок на множестве действий). Пусть задан лог событий L одного процесса P. Будем говорить, что действие B *следует* за действием A, $A \preceq_L B$, если в логе L действие B начинается после того, как заканчивается выполнение действия A, либо существует действие C такое, что C следует за A, а B следует за C.

Определение 3 (Зависимость действий). Пусть задан лог событий L одного процесса P. Будем говорить, что действие B *зависит* от действия A, $A \rightarrow_L B$, если B следует за A, а A не следует за B. Если A следует за B и B следует за A, либо A не следует за B и B не следует за A, то, будем говорить, что A и B *независимы*.

Определения частичного порядка и зависимости действий могут быть заданы и для одной трассы. Далее, если не сказано обратного, записи $A \preceq B$, $A \rightarrow B$, обозначают частичный порядок и зависимость действий в логе L.

Определение 4 (Граф зависимостей). Пусть задано множество действий V и лог L некоторого процесса, ориентированный граф G называется *графом зависимостей*, если путь из A в B в графе G существует тогда и только тогда, когда B зависит от A.

Для заданного лога может существовать несколько графов зависимостей. Два графа с одинаковым транзитивным замыканием представляют собой одинаковое множество зависимостей [19].

Можно ввести определение *подграфа G', порождаемого трассой t*, в графе зависимостей $G = (V, E)$: $G' = (V', E')$, где $V' = \{A \in t\} \subseteq V$ и $E' = \{(B, A) \in E \mid B \preceq_t A\}$, где $\preceq_t$ – отношение частичного порядка на действиях, задаваемое трассой t.

Определение 5 (Согласованность трассы с графом зависимостей). Пусть задан граф зависимостей процесса P $G = (V, E)$, трасса t. Трасса t согласуется с G, если подграф $G' = (V', E')$, порождаемый трассой t, связен; первое и последнее действия трассы t – действия начинающее и завершающее процесс P; все вершины в V' достижимы из начального действия.

Не все графы зависимостей, содержащие в себе зависимости некоторого лога L, являются корректными к рассмотрению. Для формализации ограничений, которые обычно накладываются на графы зависимостей, чтобы они могли называться моделями процесса, вводится определение *конформного графа*:

Определение 6 (Конформный граф). Граф зависимостей G называется *конформным* логу L, если выполнены следующие условия:

- Для каждой зависимости в L существует путь в G;
- Между независимыми действиями в L не существует пути в G;
- Каждая трасса лога L согласована с G.

Определение 7. Пусть задана трасса $t = A_1 \ldots A_q$. *Подстрокой трассы t длины $r \leqq q$* называется трасса $t' = A_i A_{i+1} \ldots A_{i+r-1}$, где $i = 1,\ldots,q-r+1$.

B. Постановка задачи

В работе [15] были предложены три алгоритма и рассчитана временная сложность построения модели в виде ациклического ориентированного графа, в зависимости от различных свойств лога процесса. Пусть $m = |L|$ – количество трасс в логе, $n = |V|$ – количество возможных действий процесса P. Основным предположением на входные параметры алгоритмов было предположение, что количество трасс m в логе L обычно превышает количество действий n. Данные результаты будут использоваться в данной работе в виде лемм:

Лемма 1 ([15]). *Пусть лог процесса L содержит трассы, в которых каждое действие алфавита V встречается ровно по одному разу. Сложность алгоритма построения модели процесса для лога L составляет $O(mn^2)$.*

Лемма 2 ([15]). *Пусть лог процесса L содержит трассы произвольной длины, но в каждой трассе нет повторов действий. Сложность алгоритма построения модели процесса для лога L составляет $O(mn^3)$.*

Лемма 3 ([15]). *Пусть лог процесса L содержит трассы произвольной длины, в которых могут быть повторы действий. Сложность алгоритма построения модели процесса для лога L составляет $O(m(kn)^3)$, где k*

– *максимальное количество повторов действия среди действий в логе.*

Основными этапами алгоритмов, которые строят модель процесса в виде графа по заданному логу L предложенных в работе [15], являются:

1. Добавление в граф всех дуг, соответствующих зависимостям между действиями, представленными в логе L;
2. Если между двумя вершинами есть дуги, направленные в противоположные стороны, такие дуги удаляются;
3. Удаление дуг внутри сильно-связных компонент графа;
4. Добавление меток на дуги, которые присутствуют в транзитивных замыканиях подграфов, порождаемых трассами лога L. Удаление дуг без меток;
5. Склейка вершин, соответствующих одинаковым действиям.

В настоящей работе, целью является расширение области применения вышеизложенных алгоритмов на случай, когда лог содержит события от нескольких одновременно выполняющихся процессов. Сначала рассматривается случай, когда известно, что одна трасса может содержать данные ровно одного процесса. Далее рассмотрен случай, когда в рамках одной трассы могут встречаться данные нескольких процессов.

III. Результаты

Рассматривается s процессов $P_1,\ldots,P_s$, с непустыми множествами действий $V_1,\ldots,V_s$ и графами $G_1=(V_1,E_1),\ldots,G_s=(V_s,E_s)$ соответственно.

Задача данной работы: пусть задан некоторый лог L, который может содержать в себе трассы (следы) нескольких процессов, необходимо построить конформные графы (модели) этих процессов и оценить трудоемкость данного построения.

Пусть $M = |L|$, $n = max_i|V_i|$, $i=1,\ldots,s$, k – максимальное количество повторов действия в логе L.

Теорема 1 (Процессы с различными множествами действий). Пусть задан лог L для s процессов. Пусть s процессов имеют попарно различные множества действий, $V_i \cap V_j = \{\}$, при $i \neq j$, $i, j = 1,\ldots,s$. Тогда сложность построения s конформных графов составляет $O(Mn^2)$, $O(Mn^3)$, $O(M(kn)^3)$ соответственно, в зависимости от свойств лога L.

Доказательство:

Так как множества действий для каждого из процессов не пересекаются, алгоритмы, предложенные в работе [15] не требуют модификации и будут работать корректно и на s процессах. Связано это со структурой самих алгоритмов, которая состоит в том, что на начальных шагах в граф зависимостей добавляются дуги, задающие все зависимости лога. Далее происходит редуцирование дуг получившегося графа. Тем самым, в графе зависимостей остается минимально-возможное количество дуг с сохранением необходимых зависимостей лога.

По условию теоремы, множества действий в процессах не пересекаются. Таким образом, дуги между процессами не могут быть добавлены на первом шаге алгоритмов, поэтому алгоритмы корректно сначала построят, а потом минимизируют s различных компонент связности, каждая из которых и задает граф зависимостей (модель) процесса. Поэтому сложность алгоритмов остается такой же, как в Леммах 1, 2, 3 с поправкой на увеличившийся размер лога M.

□

Если построено s конформных графов зависимостей для s процессов и известно, что в поступающей трассе t может быть след только одного процесса, то задача наличия аномалий в t сводится к задаче определения согласованности этой трассы с s графами зависимостей.

Следствие 1 (Поиск аномалий в процессах с различными множествами действий). Пусть s процессов имеют попарно различные множества действий, $V_i \cap V_j = \{\}$, при $i \neq j$, $i, j = 1,\ldots,s$. Пусть для s процессов построено s конформных графов зависимостей, тогда поиск аномалий в некоторой трассе t может быть выполнен за $O(s+|t|+n(n+e))$.

Доказательство:

Пусть $n = max_i|V_i|$, $e = max_i|E_i|$, $i=1,\ldots,s$. Пусть $G' = (V', E')$ – граф, порождаемый трассой t.

Тогда, проверка согласованности трассы t построенным конформным графам состоит из следующих шагов:

1. Определение соответствующего трассе t конформного графа: так как действия между процессами для Теоремы 1 не пересекаются, то достаточно $O(s)$ операций, чтобы сравнить, например, начальные действия построенных графов и начальное действие проверяемой трассы t. Пусть нужный конформный граф для проверки соответствия трассы t – граф $G = (V, E)$.
2. Проверка $V' \subseteq V$. При хранении в памяти вершин V, проверку можно осуществить за $O(|t|)$ операций.
3. Проверка достижимости вершин из V' из начальной вершины графа G. На этом шаге уже известно, что количество различных действий в трассе t не превышает n, поэтому проверку можно осуществить за не более n запусков алгоритма BFS [20], один запуск BFS имеет сложность $O(n+e)$, итоговая сложность проверки $O(n(n+e))$.
4. Построение графа G' для трассы t. Построение графа для одной трассы, сводится к построению сначала отношения $\leq_t$, а потом построения отношения $\rightarrow_t$, которое соответствует множеству дуг E'. Пусть хранение отношения $\leq_t$ организовано с помощью хеш-таблицы, где ключи – уже встреченные в трассе t пары действий, значения – встречалась ли пара в обратном порядке. Тогда построение отношения $\rightarrow_t$, возможно осуществить за $O(|t|)$ операций и $O(n^2)$ дополнительной памяти для хранения пар действий.
5. Проверка $E' \subseteq E$. Так как на этом шаге уже известно, что $|V'|<n$, проверка займет не более $O(n^2)$ операций.
6. Проверка связности графа G'. На этом шаге уже известно, что количество ребер не превышает e, поэтому проверку связности возможно осуществить с помощью одного запуска алгоритма BFS, сложность $O(n+e)$.

Итоговая сложность проверки согласованности трассы по вышеперечисленным шагам составляет $O(s+|t|+n(n+e)+|t|+n^2+(n+e))$, то есть $O(s+|t|+n(n+e))$.

□

Пусть $m = max_i|L_i|$, L_i – часть лога L, соответствующая i-му процессу, $i = 1,\ldots,s$. Если справедлив тот факт, что процесс однозначно можно идентифицировать его начальным действием, то, при условии s различных начальных действий, справедлива следующая теорема:

Теорема 2 (Отличимые процессы по начальному действию). Пусть задан лог L для s процессов. Пусть s процессов имеют попарно различные начальные действия. Тогда сложность построения s конформных графов составляет $O(smn^2)$, $O(smn^3)$, $O(sm(kn)^3)$ соответственно, в зависимости от свойств лога L.

Доказательство:

Так как процессы имеют попарно различные начальные действия, то за M операций, можно разбить лог L на частичные логи L_i такие, что $L_i \cap L_j = \{\}$, $i \neq j$, $i, j = 1,\ldots,s$. Для хранения соответствия начального действия процессу потребуется $O(s)$ дополнительной памяти.

Пусть $m = max_i|L_i|$, $n = max_i|V_i|$, $i = 1,\ldots,s$. Так как часть лога L_i соответствует логу одного процесса, то становятся справедливы Леммы 1, 2, 3. Учитывая, что $M \leqq sm$, будут справедливы верхние оценки:

- Лог каждого из s процессов содержит только трассы, в которых каждое действие алфавита V_i встречается ровно по одному разу. Используя Лемму 1, сложность алгоритма составит $O(smn^2)$.
- Лог каждого из s процессов содержит трассы произвольной длины, но в каждой трассе нет повторов действий. Используя Лемму 2, сложность алгоритма составит $O(smn^3)$.
- Лог каждого из s процессов содержит трассы трассы произвольной длины, в которых могут быть повторы действий. Используя Лемму 3, сложность алгоритма составит $O(sm(kn)^3)$, где k – максимальное количество повторов действия среди действий в логе.

□

Следствие 2 (Поиск аномалий в процессах, отличимых по начальному действию). Пусть s процессов имеют попарно различные начальные действия. Пусть для s процессов построено s конформных графов зависимостей, тогда поиск аномалий в некоторой трассе t может быть выполнен за $O(s+|t|+n(n+e))$.

Доказательство:

Доказательство следствия полностью повторяет доказательство Следствия 1.

□

Рассмотрим случай, когда не накладывается условие на попарно различные начальные действия для каждого из s процессов, но есть возможность разделить лог L на части по одному уникальному для процесса действию. Стоит отметить, что таким образом обеспечивается необходимое и достаточное условие существования системы различных представителей согласно теореме Ф. Холла [21]:

Теорема 3 (Отличимые процессы по некоторому действию). Пусть задан лог L для s процессов. Пусть каждый из s процессов имеет по одному известному действию A_i, $i = 1,\ldots,s$ такому, что оно не содержится в остальных процессах и содержится в каждой трассе, соответствующей i-му процессу. Тогда сложность построения s конформных графов составляет $O(smn^2)$, $O(smn^3)$, $O(sm(kn)^3)$ соответственно, в зависимости от свойств лога L.

Доказательство:

Если каждый из процессов имеет по одному известному заранее действию A_i, $i = 1,\ldots,s$ отличающему его от других, то за один проход по логу L, можно получить разбиение по логам процессов L_i таких, что $L_i \cap L_j = \{\}$, $i \neq j$, $i, j = 1,\ldots,s$.

Проход по логу не превышает $|L| \times |t_{max}|$, где t_{max} трасса лога L максимальной длины. Так как $|L| \leqq sm$, а $|t_{max}| \leqq kn$, то разбиение общего лога на логи процессов имеет оценку $O(smkn)$, что не превышает оценок на построение s конформных графов зависимостей, дальнейшее доказательство повторяет доказательство Теоремы 2.

□

Следствие 3 (Поиск аномалий в процессах, отличимых по некоторому действию). Пусть каждый из s процессов имеет по одному известному действию A_i, $i = 1,\ldots,s$ такому, что оно не содержится в остальных процессах и содержится в каждой трассе, соответствующей i-му процессу. Пусть для s процессов построено s конформных графов зависимостей, тогда поиск аномалий в некоторой трассе t может быть выполнен за $O(|t|+n(n+e))$.

Доказательство:

Доказательство следствия аналогично доказательству Следствия 1, за исключением шага 1: так как нам известны уникальные действия для процессов A_i, $i = 1,\ldots,s$, то достаточно одного прохода по трассе t, чтобы найти одно из действий A_i, тем самым выбрав нужный i-й конформный граф, сложность шага $O(|t|)$.

□

Теорема 2 рассматривает случай уникальных для процессов подстрок длины 1, которые начинаются с первой буквы трасс лога L. Справедливо обобщение этой теоремы:

Теорема 4 (Отличимые процессы по конечной подстроке). Пусть задан лог L для s процессов. Пусть известно, что каждая из трасс лога, начиная с некоторого номера i содержит подстроки длины r, каждая из которых может принадлежать только одному из s процессов. Тогда сложность построения s конформных графов составляет $O(smn^2)$, $O(smn^3)$, $O(sm(kn)^3)$ соответственно, в зависимости от свойств лога L.

Доказательство:

Пусть $h: V^r \rightarrow \{0,\ldots,s\text{-}1\}$ – идеальная хеш-функция [20]. Так как по предположению, накладываемому на лог L, количество различных слов длины r, совпадает с количеством процессов, идеальная хеш-функция h будет минимальной и может быть построена за $O(s)$, используя алгоритм из работы [22]. Тогда сложность разбиения лога L на части составляет $O(M)$, так как вычисление h от каждой из подстрок трасс лога, начинающихся с номера i длины r, в худшем случае имеет сложность $O(1)$. Для хранения отображения подстрок длины r в номера процессов потребуется $O(sr)$ дополнительной памяти.

Так как $M \leqq sm$, то сложность построения минимальной идеальной хеш-функции и сложность разбиения общего лога на логи процессов не превысит сложности построения s конформных графов

зависимостей, дальнейшее доказательство повторяет доказательство Теоремы 2.

□

Следствие 4 (Поиск аномалий в процессах, отличимых по конечной подстроке). Пусть известно, что трасса *t*, начиная с некоторого номера *i* содержит подстроку длины *r*, которая может принадлежать только одному из *s* процессов. Пусть для *s* процессов построено *s* конформных графов зависимостей, тогда поиск аномалий в трассе *t* может быть выполнен за $O(|t|+n(n+e))$.

Доказательство:

Доказательство следствия аналогично доказательству Следствия 1, за исключением шага 1: для определения нужного конформного графа необходимо использовать ту же идеальную хеш-функцию *h*, которая использовалась при построении моделей *s* процессов. Тогда сложность выполнения шага 1 составит $O(1)$.

□

Рассмотрим теперь случай, когда в логе *L* в одной трассе могут содержаться данные нескольких процессов.

Стоит заметить, что каждый из алгоритмов работы [15] после того, как были добавлены все зависимости из лога, удаляет те дуги, которые идут в обе стороны для пар действий. То есть, если согласно логу *L* существует зависимость $A \leq B$ и зависимость $B \leq A$, то дуги между вершинами, соответствующими действиям *A* и *B*, будут удалены. Если при этом действие *A* принадлежит процессу 1, а действие *B* принадлежит процессу 2, то это означает, что найдено условие, позволяющее отличать различные одновременно функционирующие процессы. Однако, это условие накладывает ограничение на "полноту" лога, чтобы каждое из действий одного процесса было независимо от каждого из действий другого процесса.

Пусть $M = |L|$, $n = max_i|V_i|$, $i = 1,\ldots,s$, k – максимальное количество повторов действия в логе *L*. Рассмотрим случай $k = 1$, где применимы Леммы 1 и 2.

Теорема 5 (Множество процессов в трассе без повторов действий). Пусть задан лог *L*, который не содержит повторов действий в рамках одной трассы. Пусть в каждой из трасс лога *L* могут присутствовать действия от 1 до *s* процессов. Если для всех действий $A \in V_i$, $B \in V_j$, $i \neq j$, $i, j = 1,\ldots,s$ в логе *L* выполнено $A \leq B$ и $B \leq A$, то сложность построения *s* конформных графов составляет $O(Mn^2)$, $O(Mn^3)$ соответственно, в зависимости от свойств лога *L*.

Доказательство:

Если для всех $A \in V_i$, $B \in V_j$, $i \neq j$, $i, j = 1,\ldots,s$ выполнено $A \leq B$ и $B \leq A$, то пары действий из разных процессов будут независимы между собой. Так как такие пары действий независимы, то дуги между подграфами, соответствующими разным процессам, будут удалены на шаге удаления двунаправленных дуг (шаг 2 основных этапов алгоритмов для одного процесса). Таким образом, при предположении, что лог одного процесса содержит достаточно информации о самом процессе, действия одного процесса будут обязательно принадлежать одной компоненте связности и образуется ровно *s* компонент связности.

В последующих шагах алгоритмов из работы [15] не происходит добавления дуг. Поэтому далее алгоритмы минимизируют построенные *s* компонент связности, каждая из которых и задает модель одного из *s* процессов. Таким образом, алгоритмы из работы [15] не требуют модификации для данного расширения и будут работать корректно для *s* процессов. Поэтому будут справедливы Леммы 1, 2, 3 с поправкой на увеличившийся размер лога.

□

Возможно рассчитать нижнюю границу количества действий в логе *L* по всем трассам для ограничений, накладываемых в Теореме 5. Пусть $M_a = \Sigma_t|t|$, $t \in L$ – общее количество действий в логе, $n_i = |V_i|$, $i = 1,\ldots,s$ – количество действий каждого из *s* процессов.

Следствие 5. Пусть задан лог *L*, который не содержит повторов действий. Пусть в каждой из трасс лога *L* могут присутствовать действия от 1 до *s* процессов. Пусть для всех действий $A \in V_i$, $B \in V_j$, $i \neq j$, $i, j = 1,\ldots,s$ в логе *L* выполнено $A \leq B$ и $B \leq A$. Тогда $M \geqq 2$, $M_a \geqq 2 \Sigma_{i=1,\ldots,s} n_i$.

Доказательство:

Оценка следует из расчета количества действий в парах вида $A \leq B$ и $B \leq A$, где $A \in V_i$, $B \in V_j$, $i \neq j$, $i, j = 1,\ldots,s$. Так как в рассматриваемых случаях $k = 1$, то трассы вида *ABA* и *BAB* будут невозможны. Поэтому для построения, например пары $A \leq B$ и $B \leq A$ потребуется 2 трассы *AB* и *BA*. Таким образом, чтобы отличить *s* процессов друг от друга потребуется как минимум 2 трассы в логе, где в первой трассе задается прямая зависимость между всеми действиями всех процессов, а во второй трассе обратная зависимость.

□

Примером достижения нижней оценки на количество трасс и действий из следствия 1 будут те процессы, где зависимости каждого из процессов возможно уместить в одну трассу. Например, пусть $V_1 = \{A, B, C\}$ и процесс p_1 может содержать только трассу *ABC*, $V_2 = \{D, E\}$ и процесс p_2 может содержать только трассу *DF*. Тогда примером лога, в котором сохраняется зависимость внутри процессов, а между процессами действия будут независимы, является лог *{ABCDE, DEABC}*.

Если построенные *s* конформных графов по Теореме 5 обладают свойствами из Теорем 1, 2, 3, 4 и некоторая трасса *t* содержит в себе след только одного процесса, то для ответа на вопрос, является ли трасса *t* аномальной, могут быть применимы Следствия 1, 2, 3, 4 соответственно.

IV. Заключение

В данной работе рассмотрена задача построения формальной модели множества процессов в виде ациклических ориентированных графов и задача поиска аномалий с помощью построенных эталонных моделей процессов. Для задачи построения моделей процесса рассмотрены случаи наличия следов нескольких процессов как между трассами одного лога событий, так и внутри трасс одного лога событий. Найдены ограничения применимости алгоритмов для одного процесса и определены оценки сложности построения формальных моделей для множества процессов в зависимости от свойств лога. Для задачи поиска аномалий предложены алгоритмы с оценками сложностями, линейными относительно длины трассы,

для которой требуется дать ответ об аномальности, и квадратичными относительно максимально возможного количества действий и зависимостей в одном моделируемом процессе. Также даны оценки на дополнительно требуемую память и минимальный размер лога, для случая, когда в рамках одной трассы встречаются данные множества процессов.

Библиография

[1] R. Sarno, F. Sinaga, K. R. Sungkono, *Anomaly detection in business processes using process mining and fuzzy association rule learning.* Journal of Big Data, 2020, vol. 7, no. 1, pp. 1-19.

[2] J. Cremerius, M. Weske, *Data-Enhanced Process Models in Process Mining.* arXiv preprint arXiv:2107.00565, 2021.

[3] R. Bhogal, A. Garg, *Anomaly Detection and Fault Prediction of Breakdown to Repair Process Using Mining Techniques*. International Conference on Intelligent Engineering and Management (ICIEM), IEEE, 2020, pp. 240-245.

[4] K. Diba et al., *Extraction, correlation, and abstraction of event data for process mining*. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 2020, vol. 10, no. 3, p. e1346.

[5] A. Pika et al., *Privacy-preserving process mining in healthcare.* International journal of environmental research and public health, 2020, vol. 17, no. 5, p. 1612.

[6] E. M. Gold, *Complexity of automaton identification from given data.* Information and control, 1978, vol. 37, no. 3, pp. 302-320.

[7] L. Miclet, *Grammatical inference.* Syntactic and Structural Pattern Recognition—Theory and Applications, 1990, pp. 237-290.

[8] W. Van der Aalst, T. Weijters, L. Maruster, *Workflow mining: Discovering process models from event logs.* IEEE Transactions on Knowledge and Data Engineering, 2004, vol. 16, no. 9, pp. 1128-1142.

[9] W. Van Der Aalst, K. M. Van Hee, K. van Hee, *Workflow management: models, methods, and systems.* MIT press, 2004.

[10] J. E. Cook, A. L Wolf, *Discovering models of software processes from event-based data.* ACM Transactions on Software Engineering and Methodology (TOSEM), 1998, vol. 7, no. 3, pp. 215-249.

[11] S. Das, M. C. Mozer, *A unified gradient-descent/clustering architecture for finite state machine induction.* Advances in neural information processing systems, 1994, pp. 19-26.

[12] Z. Zeng, R. M. Goodman, P. Smyth, *Learning finite state machines with self-clustering recurrent networks.* Neural Computation, 1993, vol. 5, no. 6, pp. 976-990.

[13] R. Agrawal, R. Srikant, *Fast algorithms for mining association rules.* Proc. 20th int. conf. very large data bases, VLDB, 1994, vol. 1215, pp. 487-499.

[14] R. Agrawal, R. Srikant, *Mining sequential patterns.* Proceedings of the eleventh international conference on data engineering, IEEE, 1995, pp. 3-14.

[15] R. Agrawal, D. Gunopulos, F. Leymann, *Mining process models from workflow logs.* International Conference on Extending Database Technology, Springer, Berlin, Heidelberg, 1998, pp. 467-483.

[16] H. Mannila, H. Toivonen, A. I. Verkamo, *Discovery of frequent episodes in event sequences.* Data mining and knowledge discovery, 1997, vol. 1, no. 3, pp. 259-289.

[17] R. Agrawal, T. Imieliński, A. Swami, *Mining association rules between sets of items in large databases.* Proceedings of the 1993 ACM SIGMOD international conference on Management of data, 1993, pp. 207-216.

[18] И. Ю. Терёхина, А. А. Грушо, Е. Е. Тимонина, С. Я. Шоргин, *Построение моделей процесса с помощью простых сетей Петри.* Системы и средства информатики, 2020, vol. 30, no. 4, pp. 61-75.

[19] A. V. Aho, M. R. Garey, J. D. Ullman, *The transitive reduction of a directed graph.* SIAM Journal on Computing, 1972, vol. 1, no. 2, pp. 131-137.

[20] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, *Introduction to algorithms.* MIT press, 2009.

[21] А. А. Грушо, Е. Е. Тимонина, Н. А. Грушо, И. Ю. Терёхина, *Выявление аномалий с помощью метаданных.* Информатика и её применения, 2020, vol. 14, no. 3, pp. 76-80.

[22] Z. J. Czech, G. Havas, G., B.S. Majewski, *An optimal algorithm for generating minimal perfect hash functions,* Information processing letters, 1992, vol. 43, no. 5, pp. 257-264.

Anomaly detection in several running processes

I.Yu. Teryokhina

***Abstract*—The paper discusses the problem of process mining with further anomalies detection for constructed process models. The algorithms extension for the case of several running processes with some restrictions on their structure has been investigated. An acyclic directed graph is considered as a formal model for a process.**

The event log traces containing the data from one and several processes were examined. The complexity problem of detecting anomalies is studied, for the cases when the processes consist of various actions sets and when the intersection of the actions sets of the processes is a finite set. The applicability limitations of the proposed algorithms' extension are found.

The complexity estimates for the formal models' construction of a set of processes problem and for anomaly detection with constructed formal models' problem are determined. For the case when data from several processes are encountered within the same trace some additional estimates are given, concerning the additional required memory and the minimum size of the log.

***Keywords*—anomaly detection, directed acyclic graph, process mining, process model construction.**

REFERENCES

[1] R. Sarno, F. Sinaga, K. R. Sungkono, *Anomaly detection in business processes using process mining and fuzzy association rule learning.* Journal of Big Data, 2020, vol. 7, no. 1, pp. 1-19.

[2] J. Cremerius, M. Weske, *Data-Enhanced Process Models in Process Mining.* arXiv preprint arXiv:2107.00565, 2021.

[3] R. Bhogal, A. Garg, *Anomaly Detection and Fault Prediction of Breakdown to Repair Process Using Mining Techniques*. International Conference on Intelligent Engineering and Management (ICIEM), IEEE, 2020, pp. 240-245.

[4] K. Diba et al., *Extraction, correlation, and abstraction of event data for process mining*. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 2020, vol. 10, no. 3, p. e1346.

[5] A. Pika et al., *Privacy-preserving process mining in healthcare.* International journal of environmental research and public health, 2020, vol. 17, no. 5, p. 1612.

[6] E. M. Gold, *Complexity of automaton identification from given data.* Information and control, 1978, vol. 37, no. 3, pp. 302-320.

[7] L. Miclet, *Grammatical inference.* Syntactic and Structural Pattern Recognition—Theory and Applications, 1990, pp. 237-290.

[8] W. Van der Aalst, T. Weijters, L. Maruster, *Workflow mining: Discovering process models from event logs.* IEEE Transactions on Knowledge and Data Engineering, 2004, vol. 16, no. 9, pp. 1128-1142.

[9] W. Van Der Aalst, K. M. Van Hee, K. van Hee, *Workflow management: models, methods, and systems.* MIT press, 2004.

[10] J. E. Cook, A. L Wolf, *Discovering models of software processes from event-based data.* ACM Transactions on Software Engineering and Methodology (TOSEM), 1998, vol. 7, no. 3, pp. 215-249.

[11] S. Das, M. C. Mozer, *A unified gradient-descent/clustering architecture for finite state machine induction.* Advances in neural information processing systems, 1994, pp. 19-26.

[12] Z. Zeng, R. M. Goodman, P. Smyth, *Learning finite state machines with self-clustering recurrent networks.* Neural Computation, 1993, vol. 5, no. 6, pp. 976-990.

[13] R. Agrawal, R. Srikant, *Fast algorithms for mining association rules.* Proc. 20th int. conf. very large data bases, VLDB, 1994, vol. 1215, pp. 487-499.

[14] R. Agrawal, R. Srikant, *Mining sequential patterns.* Proceedings of the eleventh international conference on data engineering, IEEE, 1995, pp. 3-14.

[15] R. Agrawal, D. Gunopulos, F. Leymann, *Mining process models from workflow logs.* International Conference on Extending Database Technology, Springer, Berlin, Heidelberg, 1998, pp. 467-483.

[16] H. Mannila, H. Toivonen, A. I. Verkamo, *Discovery of frequent episodes in event sequences.* Data mining and knowledge discovery, 1997, vol. 1, no. 3, pp. 259-289.

[17] R. Agrawal, T. Imieliński, A. Swami, *Mining association rules between sets of items in large databases.* Proceedings of the 1993 ACM SIGMOD international conference on Management of data, 1993, pp. 207-216.

[18] I. Yu. Teryokhina, A. A. Grusho, E. E. Timonina, S. Ya. Shorgin, *Constructing process models represented by simple Petri nets.* Systems and Means of Informatics, 2020, vol. 30, no. 4, pp. 61-75.

[19] A. V. Aho, M. R. Garey, J. D. Ullman, *The transitive reduction of a directed graph.* SIAM Journal on Computing, 1972, vol. 1, no. 2, pp. 131-137.

[20] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, *Introduction to algorithms.* MIT press, 2009.

[21] A. A. Grusho, E. E. Timonina, N. A. Grusho, I. Yu. Teryokhina, *Identifying anomalies using metadata.* Informatics and applications, 2020, vol. 14, no. 3, pp. 76-80.

[22] Z. J. Czech, G. Havas, G., B.S. Majewski, *An optimal algorithm for generating minimal perfect hash functions*, Information processing letters, 1992, vol. 43, no. 5, pp. 257-264.