Санкт-Петербургский государственный университет

Кафедра системного программирования
Программная инженерия

Лозов Петр Алексеевич

Идентификация приложения во время его исполнения в различных эксплуатационных ситуациях

Бакалаврская работа

Научный руководитель:
ст. преп. Баклановский М. В.

Рецензент:
д. т. н., профессор Зикратов И. А.

Санкт-Петербург
2016

SAINT-PETERSBURG STATE UNIVERSITY

Department of Software Engineering
Software Engineering

Petr Lozov

Identification of application during its execution in different operational situations

Graduation Thesis

Scientific supervisor:
senior lecturer Maxim Baklanovsky

Reviewer:
professor Igor Zikratov

Saint-Petersburg
2016

Оглавление

Введение

Задача обнаружения вредоносных программ – одна из важнейших задач информационной безопасности. Она включает в себя идентификацию программ, как по исходному коду, так и по особенностям поведения во время их исполнения.

Большинство антивирусных программ являются сигнатурными и решают задачу обнаружения файлов, содержащих вредоносный код. У сигнатурных антивирусных программ есть ряд недостатков: медленная реакция на новую вредоносную программу, которой нет в сигнатурной базе; игнорирование изменения поведения процесса исполнения программы.

В отличие от сигнатурных антивирусных программ, система противодействия вредоносным программам CODA[3] анализирует поведение процессов, посредством построения и сравнения *портретов* процессов – наборов часто повторяющихся последовательностей системных вызовов, которые называются *шаблонами*. В рамках этого проекта на данный момент изучены следующие вопросы[4]:

- сбор последовательностей системных вызовов потоков всех процессов,
- построение портретов процессов по собранным последовательностям системных вызовов,
- идентификация процесса в реальном времени по набору портретов.

Также исследуется задача сравнения портретов, решение которой позволит выявлять *дубликаты* – различные портреты одного процесса, возникающие при обновлении базы портретов. В настоящее время разработан алгоритм[9], решающий эту задачу. Однако при сравнении портретов на результат влияет большое количество внешних факторов, связанных с конкретной эксплуатационной ситуацией: загруженность системы, входные данные, поведение пользователя и т.д.. Это снижает схожесть различных портретов одного и того же процесса.

Одним из возможных решений данной проблемы является кластерный анализ[6], который позволяет разделить портреты на компоненты, состоящие из схожих между собой шаблонов. Данный подход позволит производить покомпонентное сравнение, минимизируя тем самым влияние компонент, возникших вследствие особенностей эксплуатационной ситуации, во время которой портрет был записан.

Целью данной работы является исследование алгоритмами кластерного анализа проблемы влияния эксплуатационной ситуации, во время которой производился сбор системных вызовов, на эффективность сравнения портретов.

Для достижения этой цели были сформулированы следующие задачи:

1) выбрать и разработать несколько различных алгоритмов кластерного анализа для шаблонов портрета;

2) сравнить эффективность алгоритмов кластерного анализа и выбрать наиболее подходящий;

3) разработать алгоритм сравнения кластеризованных портретов;

4) сравнить эффективность разработанного алгоритма и алгоритма сравнения некластеризованных портретов.

1. Обзор

1.1. CODA

CODA – это система противодействия вредоносным программам, которая разрабатывается на кафедре системного программирования математико-механического факультета Санкт-Петербургского Государственного Университета с 2009 года.

В отличие от большинства схожих систем, CODA анализирует поведение процессов, а не содержимое файлов. Неформально говоря, CODA запоминает поведение разрешенных процессов и в дальнейшем во время эксплуатации системы сравнивает поведение каждого текущего процесса со всеми разрешенными. Чем меньше общего у этих поведений, тем меньше свобод предоставляется данному процессу, вплоть до его уничтожения.

Рассмотрим более подробно, как CODA запоминает поведение разрешенных процессов. Каждый поток каждого процесса с течением времени совершает различные системные вызовы. Последовательность этих системных вызовов будем называть *следом* потока. Заметим, что каждый системный вызов однозначно описывается своим уникальным номером, поэтому след можно считать строкой над алфавитом из номеров системных вызовов. *Шаблонами* процесса назовём подстроки, которые встречаются в следе какого-либо потока процесса более одного раза. Заметим, что шаблоны могут пересекаться между собой, включаться один в другой, а также являться результатом конкатенации нескольких других шаблонов.

В процессе анализа следов всех потоков процесса, CODA формирует *портрет* процесса – некоторое подмножество достаточно длинных и достаточно часто встречающихся шаблонов процесса. Именно портрет процесса и выступает в качестве описания поведения процесса. Таким образом, для выявления подозрительной активности CODA создает портреты всех разрешенных на конкретном компьютере приложений.

Отметим, что в случае изменения поведения процесса реакция будет быстрой, так как с уменьшением похожести поведения процесса на портреты ограничения прав для этого процесса будут расти.

1.1.1. Алгоритм сравнения некластеризованных портретов

В моей курсовой работе[9] был разработан алгоритм сравнения портретов, основанный на вычислении отношения суммарной длины общих для сравниваемых портретов подстрок шаблонов и суммарной длины всех шаблонов меньшего из кластеров.

Таким образом, результатом сравнения двух портретов является число от 0 до 1, при этом, чем выше число, тем больше портреты похожи. После тестирования алгоритма была выбрана такая константа, с помощью которой различаются пары портретов процессов одного приложения, от пар портретов процессов различных приложений.

При тестировании данного алгоритма выявлено отсутствие ложных равенств портретов, однако также были обнаружены 39% ложных неравенств.

1.2. Кластерный анализ

Основное назначение кластерного анализа заключается в решении задачи разделения по каким-либо признакам некоторого набора объектов на кластеры – непересекающиеся группы объектов таким образом, чтобы объекты были схожими внутри кластеров и различались, если они из разных кластеров. Общими достоинствами алгоритмов кластерного анализа являются: возможность производить разделение объектов по произвольному количеству различных характеристик и независимость от вида анализируемых объектов, позволяющая рассматривать множества исходных объектов произвольной природы.

Все алгоритмы кластерного анализа можно разделить на следующие группы:

- иерархические,

- неиерархические.

Главная идея иерархических алгоритмов кластерного анализа исходит из предположения, что в кластеризуемом множестве присутствуют вложенные группы или, другими словами, кластеры различных порядков. Иерархические алгоритмы в свою очередь можно разделить на две группы:

- восходящие алгоритмы, которые последовательно объединяют меньшие кластеры в большие;
- нисходящие алгоритмы, которые последовательно разделяют большие кластеры на меньшие.

Неиерархические алгоритмы кластерного анализа не требуют предположения о наличии вложенных кластеров. Однако для их работы необходимо априорно определить количество кластеров, их плотность или какие-либо другие характеристики кластеров.

1.2.1. Алгоритм K-средних

Данный неиерархический алгоритм кластерного анализа является наиболее популярным, так как является наглядным и достаточно эффективным. Однако данный алгоритм требует априорное определение количества кластеров, работает только с численными векторами фиксированной длины, а также сильно зависит от выбора начальных центроидов кластера.

Выбирать начальные центроиды можно по-разному: сгенерировать случайные вектора, выбрать случайные вектора исходного множества, либо использовать более замысловатые подходы, примером которых является инициализация центроидов в алгоритме k-means++[1].

Принцип работы этого алгоритма следующий: на каждой итерации каждый кластеризуемый объект попадает в кластер, соответствующий самому ближнему центроиду, полученному на предыдущем шаге. После чего для каждого кластера вычисляется новый центроид, как среднее арифметическое значение по всем элементам этого кластера. Про-

цесс кластеризации останавливается, когда центроиды перестают значительно изменяться, а кластеры, полученные на последней итерации, считаются оптимальными.

1.2.2. Алгоритм FOREL

Неиерархический алгоритм кластерного анализа FOREL[7], предложенный Загоруйко Н. Г. и Ёлкиной В. Н. в 1967 году, основан на идеи обнаружения областей сгущения и объединения их в кластеры. Данный алгоритм в отличие от k-средних не требует априорного значения количества кластеров, однако вместо этого требует R - радиус локальных сгущений.

Так же как и k-средних работает только с численными векторами фиксированной длины.

Для выделения одного кластера выбирается случайный вектор в качестве начального центроида. Далее итеративно отбираются все вектора, расстояние от которых до центроида меньше либо равно R, после чего центроид заменяется средним арифметическим отобранных векторов. Данный процесс повторяется, пока разница между новым центроидом и центроидом, полученном на предыдущем шаге, значительна. После чего все обнаруженные для этого центроида вектора объединяются в кластер и исключаются из кластеризируемого множества. Таким образом, последовательно выделяются кластеры, пока не исчерпается кластеризируемое множество.

1.2.3. Алгоритм CLOPE

В 2002 году группой китайских ученых был предложен неиерархический алгоритм кластерного анализа под названием CLOPE[2].

Данный алгоритм, в отличие от рассмотренных выше, кластеризует множество *транзакций*, где под транзакцией подразумевается некоторый набор объектов. Например, транзакциями являются список ключевых слов в тексте и список содержащихся в конкретном блюде ингредиентов.

Основная идея алгоритма CLOPE заключается в распределении транзакций по кластерам таким образом, чтобы транзакции каждого кластера содержали как можно больше общих объектов.

Введем следующие обозначения:

- $S(C_i)$ – количество объектов во всех транзакциях кластера C_i,
- $W(C_i)$ – количество уникальных объектов во всех транзакциях кластера C_i,
- $H(C_i) = \frac{S(C_i)}{W(C_i)}$ – среднее количество повторений уникальных объектов во всех транзакциях кластера C_i.

Тогда транзакции в кластерах содержат тем больше общих объектов, чем больше значение *функции качества*:

$$profit_r(C) = \frac{\sum_{i=1}^{k} |C_i| \frac{H(C_i)}{W(C_i)^r}}{\sum_{i=1}^{k} |C_i|}$$

В данной функции качества k – количество кластеров, C_i – кластер под номером i, r – *коэффициент отталкивания*, определяющий необходимый уровень схожести транзакций внутри кластера и, следовательно, влияющий на количество кластеров.

Алгоритм CLOPE состоит из следующих шагов:

- последовательное распределение каждой транзакции в существующий или новый кластер таким образом, чтобы функция качества имела максимальное значение;
- итеративный перебор транзакций и перемещение в существующий или новый кластер так, чтобы функция качества максимально увеличилась.

Процесс перемещения транзакций прекращается, когда перемещение любой транзакции не увеличивает значение функции качества.

Отметим, что результат кластеризации зависит не только от параметра отталкивания, но и от порядка перебора транзакций.

1.2.4. Генетический алгоритм

Генетический алгоритм[5] – алгоритм поиска оптимального решения, основанный на селекции лучших элементов, подобно процессу естественного отбора в природе. К примеру, с помощью данного алгоритма можно максимизировать (или минимизировать) какую-либо целевую функцию.

Термины, используемые для описания генетического алгоритма, заимствованы из генетики:

- *особь* – объект, являющийся потенциальным решением;
- *поколение* – набор особей на некоторой итерации алгоритма;
- *функция селекции* – функция, генерирующая новую особь на основе других особей;
- *функция мутации* – функция, вносящая незначительные изменения в структуру особи.

Данный алгоритм состоит из двух шагов: инициализация первого поколения и итеративное создание новых поколений, с помощью функции селекции, мутации и отбора в следующее поколение особей, на которых достигаются наилучшие значения целевой функции. Алгоритм завершается, когда значение целевой функции перестает значительно изменяться в течение нескольких поколений.

Отметим, что при наличии функции качества кластеризации задачу кластерного анализа можно решить с помощью генетического алгоритма. Таким образом, генетический алгоритм можно считать неиерархическим алгоритмом кластерного анализа.

1.2.5. Восходящий иерархический алгоритм

Данный алгоритм, как следует из названия, является иерархическим и восходящим. Он автоматически определяет количество кластеров, а также не накладывает ограничений на природу кластеризуемых объектов, однако ему необходима функция схожести двух кластеров, а

также пороговое значение, определяющее необходимость объединения этих кластеров.

Изначально каждый кластеризуемый объект считается отдельным кластером, после чего вычисляется значение функции схожести для всех пар кластеров. Далее на каждой итерации среди всех посчитанных значений функции схожести ищется наибольшее значение. Если это значение не превосходит пороговое значение, то работа алгоритма завершена. Иначе соответствующие тому значению кластеры объединяются, производится вычисление функции схожести для нового кластера и всех остальных кластеров, и алгоритм переходит к следующей итерации.

Этот алгоритм имеет квадратичную зависимость времени от размера кластеризуемого множества: на первом шаге вычисляется расстояние между всеми парами начальных кластеров, количество которых равняется размеру кластеризируемого множества; на каждом следующем шаге алгоритм линеен, однако количество шагов в худшем случае совпадает с размером кластеризируемого множества.

2. Реализация алгоритмов кластерного анализа

Для решения задачи кластеризации портрета процесса, прежде всего, необходимо выбрать наиболее подходящий алгоритм кластерного анализа. Для этого были разработаны несколько алгоритмов с целью сравнения их эффективности применительно к различным портретам процессов.

Отметим, что в описанных ниже алгоритмах используется алгоритм сравнения некластеризованных портретов для вычисления схожести кластеров. Данное использование корректно, так как любые два кластера также можно считать портретами того же процесса.

2.1. Алгоритм K-средних

При попытке применить данный алгоритм к портрету процесса возникает две проблемы: элементы портрета – шаблоны, не являющиеся численными векторами фиксированной длины, и количество кластеров заранее неизвестно.

В качестве векторного представления шаблонов были использованы вектора частот *n-грамм* – подстрок длины N. Отметим, что корректными значениями N являются целые числа больше 0 и меньше либо равные 5. Ограничение сверху возникает вследствие того факта, что минимальная длина шаблона равняется 5. С ростом N длина вектора растет экспоненциально. К примеру, если бы в портрете присутствовало 100 различных номеров системных вызовов, то количество всех различных 5-грамм равнялось бы 10^{10}, и размер вектора частот 5-грамм, состоящего из чисел с плавающей запятой, составлял бы более 37 гигабайт. Однако количество n-грамм в шаблоне длины K равняется $K - n + 1$. Значит в векторе частот количество значений, не равных нулю, не превосходит длины шаблона, что позволяет сильно сжать вектор частот, представив его в виде словаря, где ключом является n-грамма и значением – частота, не равная нулю.

Для решения проблемы определения количества кластеров рассматривались два подхода.

1. Выполнение кластеризации для различного количества кластеров и выбор наиболее качественного результата.

2. Преобразование алгоритма k-средних в иерархический нисходящий алгоритм, не требующий априорного значения количества кластеров.

Однако первый подход требует слишком больших временных затрат, так как необходимо много раз производить кластеризацию, временна́я сложность которой растет с увеличением количества кластеров.

Второй же подход заключается в итеративном разделении кластеров исходным алгоритмом k-средних на два кластера пока это возможно. Данный подход также требует большого количества итераций, однако, во-первых, количество кластеров для каждой итерации равняется двум, во-вторых, размеры кластеризуемых множеств уменьшаются.

Но при таком подходе возможно возникновение фиктивных кластеров, между которыми нет значительного различия. Фиктивные кластеры необходимо обнаружить и объединить. Для этого необходима функция схожести двух кластеров. В качестве подобной функции был использован алгоритм сравнения некластеризованных портретов.

Таким образом, обе проблемы решены. В результате получен нисходящий иерархический алгоритм k-средних, принимающий параметрами длину n-грамм для преобразования шаблонов в численные вектора и значение минимального размера кластера для определения момента завершения работы алгоритма.

Подводя итог, приводим краткое описание работы нисходящего иерархического алгоритма k-средних.

Подготовка:

- преобразование шаблонов портретов в вектора частот n-грамм.

Итерация:

- поиск необработанного кластера достаточного размера, если таковой не найден, то переход к завершению;
- разделение кластера на две части с помощью алгоритма k-средних несколько раз;
- выбор результата, в котором кластеры наименее похожи;
- если оба кластера достаточного размера, то замена исходного кластера на полученные, иначе добавление метки "обработан" исходному кластеру;
- переход к следующей итерации.

Завершение:

- пока присутствуют фиктивные кластеры (которые обнаруживаются с помощью алгоритма сравнения некластеризованных портретов), производится объединение этих кластеров.

2.2. Алгоритм FOREL

Алгоритм FOREL, так же как и k-средних работает только с численными векторами фиксированной длины, поэтому необходимо предварительно преобразовать шаблоны в вектора частот n-грамм, после чего выполняется сам алгоритм FOREL.

Преобразование шаблонов в вектора приводит к потере информации (невозможно однозначно по вектору восстановить шаблон, из которого этот вектор был получен), поэтому возможно возникновение фиктивных кластеров, которые так же, как и в случае алгоритма k-средних, можно обнаружить с помощью алгоритма сравнения некластеризованных портретов и объединить.

2.3. Алгоритм CLOPE

Как было сказано в обзоре, алгоритм применяется ко множеству транзакций. Следовательно, чтобы применить алгоритм CLOPE к порт-

рету, необходимо преобразовать шаблоны портрета в транзакции. Для этого в соответствие каждому шаблону построим набор уникальных n-грамм, содержащихся в этом шаблоне (по аналогии с преобразованием шаблонов в численные вектора).

Так как результат алгоритма CLOPE зависит от того, как упорядочены транзакции, перед запуском алгоритма полученные из шаблонов транзакции переставляются случайным образом. За счет этого увеличивается вариативность алгоритма, ведь его можно запустить несколько раз для различных перестановок транзакций, а затем выбрать наиболее подходящий результат.

После завершения кластеризации необходимо выявить и объединить все фиктивные кластеры, по тем же причинам, что и в алгоритме FOREL.

2.4. Генетический алгоритм

Сначала рассмотрим алгоритм кластеризации портрета процесса на два кластера, использующий генетический алгоритм. Для этого необходимо определить те сущности, которыми оперирует генетический алгоритм.

Произвольное разбиение портрета на два кластера представим в виде битового массива, где каждый бит описывает принадлежность соответствующего ему шаблона к тому или иному кластеру. Данный вектор будет потенциальным решением, другими словами, особью.

В качестве целевой функции будем использовать алгоритм сравнения некластеризованных портретов. При минимизации данной функции достигается минимальное сходство кластеров.

Функция селекции принимает две особи и создает потомка, в котором значение некоторого бита равняется значению соответствующего бита исходных особей, если это значение у них совпадает. В противном случае, значение выбирается случайно. Отметим, что функцию селекции необходимо применять как к самим особям, так и к одной исходной, а другой – инвертированной (каждый бит заменяется на противополож-

ный). Поэтому каждая пара особей будет порождать два потока.

Функция мутации с небольшой вероятностью заменяет произвольный бит противоположным значением.

Генерация первого поколения заключается в создании битовых массивов, заполненных случайными значениями.

Таким образом, все необходимое для кластеризации портрета процесса на два кластера с помощью генетического алгоритма готово.

Для расширения данного алгоритма на произвольное количество кластеров было рассмотрено два варианта.

1. Использование массивов целых чисел, вместо битовых массивов, позволяющее описать разбиение на произвольное число кластеров.

2. Преобразование алгоритма в нисходящий иерархический алгоритм, который самостоятельно определяет количество кластеров.

Однако первый вариант экспоненциально увеличивает множество особей относительно количества кластеров, что очень сильно снижает скорость сходимости алгоритма. Поэтому был выбран второй вариант.

В итоге, генетический алгоритм кластеризации портрета, как и нисходящий алгоритм k-средних, последовательно разделяет кластеры, пока значение целевой функции мало и кластеры достаточно большие. Параметрами он принимает пороговое значение целевой функции и минимальный размер кластера.

При работе данного алгоритма возможно возникновение фиктивных кластеров, которые выявляются с помощью алгоритма сравнения некластеризованных портретов, после чего объединяются.

2.5. Восходящий иерархический алгоритм

Прежде всего, рассмотрим восходящий иерархический алгоритм, описанный в соответствующем разделе обзора. Данный алгоритм не накладывает никаких ограничений на кластеризируемые объекты, потому достаточно определить функцию схожести между кластерами, в

качестве которой будем использовать алгоритм сравнения некластеризованных портретов.

Однако данный алгоритм квадратичен по времени относительно размера кластеризируемого множества, поэтому он был изменен для снижения временных затрат.

Вместо вычисления схожести между всеми кластерами, будем последовательно выбирать пару случайных кластеров и объединять их, если они достаточно похожи. Для того чтобы сначала объединять более похожие пары, введем плавающее пороговое значение, которое с каждой следующей итерацией линейно убывает от 1 до порогового значения, которое использовалось в исходном алгоритме. Кластеры, схожесть которых ниже плавающего порогового значения, необходимо запомнить, чтобы, как только пороговое значение достаточно уменьшится, объединить эти кластеры.

3. Выбор наиболее эффективного алгоритма кластерного анализа

При выборе алгоритма кластерного анализа критериями эффективности были выбраны: количество кластеров, являющихся результатом работы алгоритма, а также отсутствие фиктивных кластеров, где под фиктивными кластерами подразумеваются кластеры с высоким уровнем сходства, выявленным с помощью алгоритма сравнения некластеризованных портретов. Однако для всех реализованных алгоритмов кластерного анализа верно, что их результат не содержит фиктивных кластеров: иерархический восходящий алгоритм объединяет фиктивные кластеры, пока таковые присутствуют; остальные алгоритмы последним шагом делают то же самое. Таким образом, основным критерием эффективности стало количество полученных кластеров.

Для того чтобы сравнить значения количеств кластеров, выделяемых разработанными алгоритмами из портретов, было собрано 20 портретов различных приложений. Сбор следов для каждого портрета проводился в течение 20 минут, причем поведение пользователя в этих приложениях никак не регламентировалось. Далее, все портреты были кластеризованы каждым разработанным алгоритмом по 10 раз. Отметим, что все параметры этих алгоритмов были подобраны имперически таким образом, чтобы максимизировать количество кластеров.

В таблице 1 представлены результаты работы алгоритмов кластерного анализа для 5 портретов из 20 обработанных. Каждая ячейка таблицы содержит максимальное количество кластеров за 10 запусков, а также округленное среднее количество, которое указано в скобках.

Как видно из данной таблицы наиболее эффективным по максимальному количеству кластеров за 10 запусков оказался алгоритм FOREL. По среднему же количеству кластеров лидирует восходящий иерархический алгоритм.

Отметим, что алгоритм CLOPE, также показавший хороший результат, оказался в несколько раз быстрее остальных алгоритмов, однако скорость не являлась рассматриваемым критерием.

	Explorer	Opera	Mine Sweeper	Task Manager	Paint W7
k-средних	38 (23)	66 (47)	16 (7)	12 (8)	17 (10)
FOREL	42 (30)	68 (52)	16 (10)	19 (8)	17 (9)
CLOPE	42 (28)	67 (48)	16 (9)	19 (7)	14 (11)
Иерархический алгоритм	40 (36)	67 (63)	16 (12)	16 (16)	17 (13)
Генетический алгоритм	3 (2)	12 (7)	1 (1)	6 (4)	5 (2)

Таблица 1: Результаты работы алгоритмов кластерного анализа

В итоге, наиболее эффективным алгоритмом кластерного анализа (применительно к портретам процессов) был признан алгоритм FOREL, который необходимо применить к портрету несколько раз и выбрать результат с наибольшим количеством кластеров.

4. Реализация алгоритма сравнения кластеризованных портретов

Теперь, когда выбран алгоритм кластеризации, мы можем для сравнения портретов сначала кластеризовать оба портрета, а затем их покомпонентно сравнить алгоритмом сравнения некластеризованных портретов. В результате мы получим *матрицу сходства*, содержащую результаты сравнения всех пар кластеров, причем как строки, так и столбцы отсортируем в порядке убывания размеров кластеров. Обозначим элементы этой матрицы, как $V_{i,j}$, где i – номер соответствующего кластера первого портрета, j – номер соответствующего кластера второго портрета.

Полученная матрица описывает схожесть кластеров, однако для сравнения самих портретов необходимо из матрицы сходства извлечь *критерий равенства*, отличающий случаи сравнения портретов различных приложений от случаев сравнения портретов одного приложения.

Хорошим критерием равенства являлась бы численная характеристика, которая близка к нулю, если сравниваемые портреты построены по следам различных процессов. И близка к единице в противоположном случае. К примеру, результат алгоритма сравнения некластеризованных портретов является приближением этой характеристики, однако с достаточно большой погрешностью.

Имперически на роль такого приближения были рассмотрены:

1) $V_{1,1}$ – результат сравнения наибольших кластеров;

2) $\sum_{i=1}^{N}\sum_{j=1}^{M}\frac{V_{i,j}}{NM}$ – среднее значение по всем элементам матрицы схожести, где N – количество кластеров первого портрета, M – количество кластеров второго портрета;

3) $\sum_{i=1}^{N}\sum_{j=1}^{M}\frac{V_{i,j}|C_i^1||C_j^2|}{|C^1||C^2|}$ – взвешенное среднее значение по всем элементам матрицы схожести, где C_i^1 – кластер под номером i первого портрета C^1, где C_j^2 – кластер под номером j второго портрета C^2;

4) $\sum_{i=1}^{N}\max_{j=1}^{M}\frac{V_{i,j}}{N}$ – усредненное наибольшее сходство для первого

портрета, а также аналогичная характеристика для второго портрета;

5) $\sum_{i=1}^{N} \max_{j=1}^{M} \frac{V_{i,j}|C_i^1|}{|C^1|}$ – взвешенное среднее значение наибольших величин для первого портрета, а также аналогичная характеристика для второго портрета;

6) $\sum_{i \in A} \sum_{j \in B} \frac{V_{i,j}}{|A||B|}$ – среднее значение по различным подматрицам матрицы схожести;

7) $\sum_{i \in A} \sum_{j \in B} \frac{V_{i,j}|C_i^1||C_j^2|}{(\sum_{i \in A}|C_i^1|)(\sum_{j \in B}|C_j^2|)}$ – взвешенное среднее значение по различным подматрицам матрицы схожести.

Для выяснения, являются ли вышеперечисленные характеристики хорошим приближением критерия равенства, был проделан следующий эксперимент.

- Было построено по 10 портретов 10 различных процессов: explorer.exe (Explorer), mspaint.exe (Paint), wmplayer.exe (Windows Media Player), iexplore.exe (Internet Explorer), winword.exe (Microsoft Word), skype.exe (Skype), mip.exe (Math Input Panel), taskmgr.exe (Windows Task Manager) AcroRd32.exe (Adobe Acrobat Reader DC) и firefox.exe (Firefox). Следы для построения этих портретов записывались группами (по 5 приложений) в течение 60 минут.

- Для каждой пары портретов были посчитаны матрицы схожести, а также прочие значения, необходимые для вычисления исследуемых характеристик.

- Матрицы схожести были преобразованы в вектора $X_{i,j}$, состоящие из значений исследуемых характеристик.

- В соответствие каждому вектору $X_{i,j}$ было добавлено значение $Y_{i,j}$, равное единице, если портреты под номерами i и j образованы от одного процесса, иначе равное нулю ($Y_{i,j}$ описывает идеальный критерий равенства).

- К всем векторам $X_{i,j}$ и значениям $Y_{i,j}$ был применен линейный метод наименьших квадратов[8]. Результатом данного алгоритма является вектор R^0, посчитанный таким образом, что функция $error(R) = \sqrt{\sum_{i=1}^{100}\sum_{j=i+1}^{100}((R, X_{i,j}) - Y_{i,j})^2}$ достигает минимума в значении R^0. Другими словами, скалярное произведение $(R^0, X_{i,j})$ минимально откланяется от $Y_{i,j}$ в квадратическом смысле. При этом, чем больше по модулю значение R^0_k, тем больший вклад вносит исследуемая характеристика под номером k в это скалярное произведение.

Наибольший вклад внесли следующие характеристики: взвешенное среднее значение наибольших величин, как для первого, так и для второго портрета (в списке характеристик расположено под номером 5), а также взвешенные средние значения по подматрицам размера 2x2 и выше для первых пяти наибольших кластеров (под номером 7). Отметим, что коэффициенты из R^0 для этих характеристик положительны, из чего следует, что каждая из них являются приближением идеального критерия равенства.

Полученные результаты можно интерпретировать следующим образом: отличительной чертой первого типа характеристик является игнорирование непохожих пар кластеров; второй тип выделяется игнорированием кластеров малого размера.

Теперь необходимо преобразовать выявленные характеристики в единое приближение критерия равенства, для которого случай ложного равенства портретов маловероятен. Это необходимо, так как основным применением алгоритма сравнения портретов является выявление дубликатов, а в случае ложного равенства портреты различных процессов будут признаны дубликатами и впоследствии объединены.

Для этого из выявленных характеристик были исключены те, которые для некоторой пары портретов различных приложений имеет большое значение. Более формально, были исключены характеристики, значение которых для некоторой пары портретов под номерами i и j превосходит максимальное допустимое значение схожести портретов различных приложений, равное 0.25, и одновременно $Y_{i,j} = 0$. Таким об-

разом, были исключены взвешенные средние значения по подматрицам размера 2x2.

Для оставшихся характеристик верны два утверждения: с одной стороны, каждая из них хорошо приближает идеальный критерий равенства, с другой, ни одна из них не приводит к ложному равенству портретов, другими словами, всякое обнаруженное с помощью любой из характеристик равенство портретов является верным. Поэтому в качестве единого приближения критерия равенства будем использовать максимальное значение по всем оставшимся характеристикам, ведь ложные равенства и для этой характеристики маловероятны, а всякое верное равенство, выявленное любой характеристикой, будет выявлено и их максимальным значением.

Таким образом, алгоритм сравнения кластеризованных портретов состоит из следующих шагов:

- вычисление q_0 – результата алгоритма сравнения некластеризованных портретов;
- выполнение кластеризации портретов;
- вычисление матрицы схожести параллельно в нескольких потоках;
- вычисление $q_1 = \sum_{i=1}^{N} \max_{j=1}^{M} \frac{V_{i,j}|C_i^1|}{|C^1|}$;
- вычисление $q_2 = \sum_{j=1}^{M} \max_{i=1}^{N} \frac{V_{i,j}|C_j^2|}{|C^2|}$;
- вычисление $q_{A,B} = \sum_{i \in A} \sum_{j \in B} \frac{V_{i,j}|C_i^1||C_j^2|}{(\sum_{i \in A}|C_i^1|)(\sum_{j \in B}|C_j^2|)}$, где $A, B \in F, F = \{X | X \in 2^{\{1,2,3,4,5\}} \, and \, |X| \geq 3\}$;
- результат алгоритма: $\max\left(q_0, q_1, q_2, \max_{A \in F, B \in F}(q_{A,B})\right)$.

5. Сравнение эффективности алгоритма сравнения некластеризованных портретов и алгоритма сравнения кластеризованных портретов

В моей курсовой работе было определено максимальное допустимое значение схожести портретов различных приложений, приблизительно равное 0.25. Данная характеристика была необходима для выявления различных портретов одного приложения.

Для выявления аналогичной характеристики были вычислены уровни сходства всех пар 20 портретов различных приложений (эти же портреты использовались для выбора наиболее эффективного алгоритма кластерного анализа). В результате максимальное допустимое значение схожести портретов различных приложений для алгоритма сравнения кластеризованных портретов оказалось приблизительно равно 0.31.

Прежде всего, качество результатов исследуемых алгоритмов необходимо было сравнить на портретах различных процессов. Для этого использовались два набора, каждый из которых содержит по 5 портретов приложений: Paint W7, Note Pad ++, WinRAR, VS 2012 и Word Pad. Построение каждого портрета проводилось в течение 40 минут, причем поведение пользователя в приложениях никак не регламентировалось. Далее, каждый портрет из первого набора сравнивался со всеми портретами из второго набора.

Результаты сравнения для обоих алгоритмов отображены в таблице 2. В этой и последующих таблицах для каждой пары портретов первым значением указан результат алгоритма сравнения некластеризованных портретов, вторым – результат алгоритма сравнения кластеризованных портретов, а также для удобства результаты умножены на 100 и округлены до целых чисел. Ошибочные значения алгоритма сравнения некластеризованных портретов выделены красным цветом, ошибочные значения алгоритма сравнения кластеризованных портретов выделены синим цветом.

	Note Pad ++		WinRAR		Paint W7		VS 2012		Word Pad	
Note Pad ++	59	59	6	6	2	9	2	5	1	2
WinRAR	5	6	40	63	3	7	3	6	4	5
Paint W7	5	8	4	5	47	67	2	4	15	16
VS 2012	2	7	2	5	2	4	50	52	2	2
Word Pad	2	8	4	4	12	15	2	5	12	42

Таблица 2: Результаты сравнения портретов различных приложений

Ошибочный результат получился только при сравнении портретов приложения Word Pad алгоритмом сравнения некластеризованных портретов. Это обусловлено тем, что поведение пользователя при построении портретов сильно различалось. Поэтому было решено проверить, насколько поведение пользователя в приложении отражается на построенном портрете этого приложения. Для этого были построены два набора по пять портретов приложения Word Pad. Данные портреты строились в течение пяти минут, причем на протяжении этого промежутка времени для первых четырёх портретов обоих наборов пользователь повторял ровно одно действие, своё для каждой пары портретов, а именно: просмотр текста, вставка теста, вставка изображения и выделение текста курсивом. Для последней пары портретов выполнялись все эти действия поочерёдно. Результаты сравнения этих двух наборов представлены в таблице 3.

Как видно из данной таблицы, поведение пользователя в приложении Word Pad оказывает очень сильное влияние на содержимое портретов. Сходства между портретами Word Pad с помощью алгоритма сравнения некластеризованных портретов не были обнаружены в 18 случаях из 25. Однако алгоритм сравнения кластеризованных портретов показал себя значительно лучше, ошибившись лишь в 5 случаях из 25. Отметим, что наиболее естественными портретами являются те, во

	Просмотр		Вставка текста		Изображение		Курсив		Всё вместе	
Просмотр	49	66	7	37	9	47	6	41	12	78
Вставка текста	5	21	51	64	2	39	2	38	19	74
Изображение	5	17	2	12	94	97	2	18	21	76
Курсив	5	43	3	45	2	16	99	99	99	99
Всё вместе	8	57	22	62	15	71	71	97	97	99

Таблица 3: Результаты сравнения портретов приложения Word Pad

время создания которых пользователь периодически менял свое поведение. Информация об этих портретах отражена в правом столбце и нижней строке. Для этих портретов алгоритм сравнения некластеризованных портретов ошибся в 6 из 10 случаях, а алгоритм сравнения кластеризованных портретов ни разу не ошибся.

Для более сложных приложений поведение пользователя накладывает гораздо меньший отпечаток на портрет, что демонстрируется в аналогичном сравнении, но для приложения Opera, представленном в таблице 4. Алгоритм сравнения некластеризованных портретов выявил

	Google		Wikipedia		Google map		YouTube		Всё вместе	
Google	75	93	53	78	13	79	23	76	51	88
Wikipedia	51	88	71	85	14	81	20	82	36	85
Google map	17	89	14	77	61	93	14	87	21	83
YouTube	25	76	29	83	11	80	54	92	61	92
Всё вместе	46	95	41	91	22	81	58	93	69	95

Таблица 4: Результаты сравнения портретов приложения Opera

сходство большинства пар портретов, однако и для этого приложения некоторые поведения сильно отличаются от остальных, что следует из наличия 11 ошибок из 25. В свою очередь, алгоритм сравнения кластеризованных портретов верно определил сходство портретов во всех 25 случаях.

Подводя итоги данного сравнения, выделим главные обнаруженные особенности:

- результат сравнения как некластеризованных, так и кластеризованных портретов различных приложений практически не зависит от поведения пользователя в процессе сбора следа;
- результат сравнения некластеризованных портретов простого приложения сильно зависит от поведения пользователя в процессе сбора следа;
- результат сравнения кластеризованных портретов простого приложения в некоторых случаях зависит от поведения пользователя;
- результат сравнения некластеризованных портретов сложного приложения в некоторых случаях зависит от поведения пользователя;
- результат сравнения кластеризованных портретов сложного приложения практически не зависит от поведения пользователя.

Заключение

В ходе данной работы были получены следующие результаты:

1) выбраны и разработаны 5 различных алгоритмов кластерного анализа для шаблонов портрета;

2) выбран наиболее эффективный алгоритм кластерного анализа FOREL;

3) разработан алгоритм сравнения кластеризованных портретов;

4) количество ложных неравенств портретов при работе алгоритма сравнения кластеризованных портретов понизилось на 71% относительно алгоритма сравнения некластеризованных портретов при полном отсутствии ложных равенств портретов;

5) данная работа была представлена на конференции СПИСОК 2016.

Список литературы

[1] Arthur David, Vassilvitskii Sergei. k-means++: the advantages of careful seeding // Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms. — 2007.

[2] Yang Yiling, Guan Xudong, You Jinyuan. CLOPE: A Fast and Effective Clustering Algorithm for Transactional Data. — 2002. — URL: `http://www.inf.ufrgs.br/~alvares/CMP259DCBD/clope.pdf` (дата обращения: 09.05.2016).

[3] Баклановский М.В, Ханов А.Р. CODA – новая система компьютерной безопасности: обзор архитектуры системы // Материалы секции 22, XXXVIII Академические чтения по космонавтике. — 2014. — С. 649–650.

[4] Баклановский М.В, Ханов А.Р. Поведенческая идентификация программ. Моделирование и анализ информационных систем. // Моделирование и анализ информационных систем. Ярославль, Том 21, Номер 6. — 2014. — Р. 120–130.

[5] Гладков Л.А., Курейчик В.В., Курейчик В.М. Генетические алгоритмы: Учебное пособие. — Москва: Физматлит, 2006.

[6] Дюран Б., Оделл П. Кластерный анализ. — Москва: Статистика, 1977.

[7] Загоруйко Н.Г. Прикладные методы анализа данных и знаний. — Новосибирск: ИМ СО РАН, 1999. — С. 38–39.

[8] Линник Ю.В. Метод наименьших квадратов и основы математико-статистической теории обработки наблюдений. — Москва: ФИЗМАТЛИТ, 1958.

[9] Лозов П.А. Метрика в пространстве портретов процессов. — 2015. — URL: `http://se.math.spbu.ru/SE/YearlyProjects/spring-2015/371/371-Lozov-report.pdf` (дата обращения: 09.05.2016).