10. Mnogofunktsionalnyy tsentr [Multifunctional centre]. *Sayt «Vikipediya»* [«Wikipedia»], 2016. Available at: https://ru.wikipedia.org/wiki/Многофункциональный_центр (accessed 26.12.2016).

11. Moroz S. Ctrategicheskoe videnie – obyazatelnoe usloviye dlya sozdaniya SOA [Strategic vision is a prerequisite for SOA]. *Informatsionnoe agentstvo «Bankir.Ru»* [News Agency "Bankir.Ru"], 2008. Available at: http://bankir.ru/publikacii/20080922/ctrategicheskoe-videnie--obyazatelnoe-ysloviya-dlya-sozdaniya-soa-1371781/ (accessed 26.12.2016).

12. Odno okno [One window]. *Sayt «Vikipediya»* [«Wikipedia»], 2015. Available at: https://ru.wikipedia.org/wiki/Одно_окно (accessed 26.12.2016).

13. Portal gosudarstvennykh uslug Rossiyskoy Federatsii [The portal of state services of the Russian Federation]. *Sayt «Vikipediya»* [«Wikipedia»], 2016. Available at: https://ru.wikipedia.org/wiki/Портал_государственных_услуг_ Российской_Федерации (accessed 26.12.2016).

14. Skobleva E. I., Sharipova A. Z. Modeli i mekhanizmy vybora optimalnoy strategii reform v sisteme vysshego obrazovaniya Rossii [Models and mechanisms of the choice of optimum strategy of reforms in system of the higher education of Russia]. *Prikaspiyskiy zhurnal: upravlenie i vysokie tekhnologii* [Caspian Journal: Control and High Technologies], 2015. no. 4, pp. 45–64

15. Tikhomirov V. P. Kachestvennoe obrazovanie dlya vsekh kak osnova formirovaniya obshchestva znaniy [Quality education for all as a basis for the formation of a knowledge society]. *Informatsionnoe obshchestvo* [Information society], 2005, no. 4, pp. 6–10.

16. Torshin D. V. *Metody integratsii dannykh kompyuternykh sistem na osnove universalnogo formata obmena dannymi* [Methods of data integration of computer systems based on the universal data exchange format], Ufa, Ufa State Aviation Technical University Publ. House, 2009. 134 p.

17. Finister James. Service Integration: Sourcing for the Future. *At Your Service*, October 2012, vol. 2, issue 3.

18. Oracle SOA Suite. *Sayt «Oracle»*. Available at: http://www.oracle.com/us/products/middleware/soa/suite/overview/index.html (accessed 26.12.2016).

УДК 004.023

МЕТОД СОКРАЩЕНИЯ ВРЕМЕНИ ОБУЧЕНИЯ АНТИВИРУСНОГО ЭВРИСТИЧЕСКОГО КЛАССИФИКАТОРА, ОСНОВАННЫЙ НА ИСПОЛЬЗОВАНИИ АЛГОРИТМА РАСШИРЕННОГО БИНАРНОГО ПОИСКА

Статья поступила в редакцию 30.12.2016, в окончательном варианте – 04.02.2017.

Демина Раиса Юрьевна, ассистент, Астраханский государственный университет, 414056, Российская Федерация, г. Астрахань, ул. Татищева, 20а, e-mail: raisapereverzeva@gmail.com

Ажмухамедов Искандар Маратович, доктор технических наук, доцент, Астраханский государственный университет, 414056, Российская Федерация, г. Астрахань, ул. Татищева, 20а, e-mail: iskander_agm@mail.ru

Гурская Татьяна Геннадиевна, кандидат технических наук, доцент, Астраханский государственный университет, 414056, Российская Федерация, г. Астрахань, ул. Татищева, 20а. e-mail: gurskai@mail.ru

Основным механизмом антивирусного распознавания вредоносных файлов является сигнатурный анализ. Однако он не способен противостоять угрозам «нулевого дня», т.е. вирусам, которые еще не были изучены антивирусными экспертами и не были добавлены в базы данных сигнатур антивирусных программ. Для противодействия таким новым вирусам применяется эвристический анализ. Его частным случаем является статический эвристический анализ. В основе эвристического анализа лежит задача бинарной классификации, которая предусматривает два этапа: обучение и распознавание. На этапе обучения из известных вирусов извлекаются признаки, на основе которых строится классификатор. Этот классификатор, в рамках использования антивирусного пакета, установленного на ЭВМ пользователя, может защитить пользователя от вирусов «нулевого дня». Наибольшие затраты времени при обучении классификатора приходятся на этап отбора признаков файла (характеризующих его как вредоносный или доброкачественный) и их вставка в общий перечень признаков. В статье проанализированы несколько способов решения указанных задач для этого этапа, выявлены их недостатки. Предложен альтернативный метод в виде алгоритма расширенного бинарного поиска, который преобразует произвольный перечень признаков в отсортированную последовательность неповторяющихся элементов. Произведена оценка сложности предлагаемого алгоритма, рассмотрены лучший и худшие случаи в отношении объемов вычислительных операций, реализующих алгоритм. Эффективность предложенного подхода подтверждена в результате его проверки с использованием ряда вычислительных экспериментов.

Ключевые слова: антивирусный эвристический анализ, сортировка, бинарный поиск, линейный поиск, сложность алгоритмов, машинное обучение, битовые карты, сортировка простыми вставками, бинарная классификация, обучение классификатора

THE METHOD OF TIME REDUCTION DURING ANTI-VIRUS HEURISTIC QUALIFIER LEARNING BASED ON THE EXPANDED BINARY SEARCH ALGORITHM USAGE

The article has been received by editorial board 30.12.2016, in the final version – 04.02.2017.

Demina Raisa Yu., Assistant, Astrakhan State University, 20a Tatishchev St., Astrakhan, 414056, Russian Federation, e-mail: raisapereverzeva@gmail.com

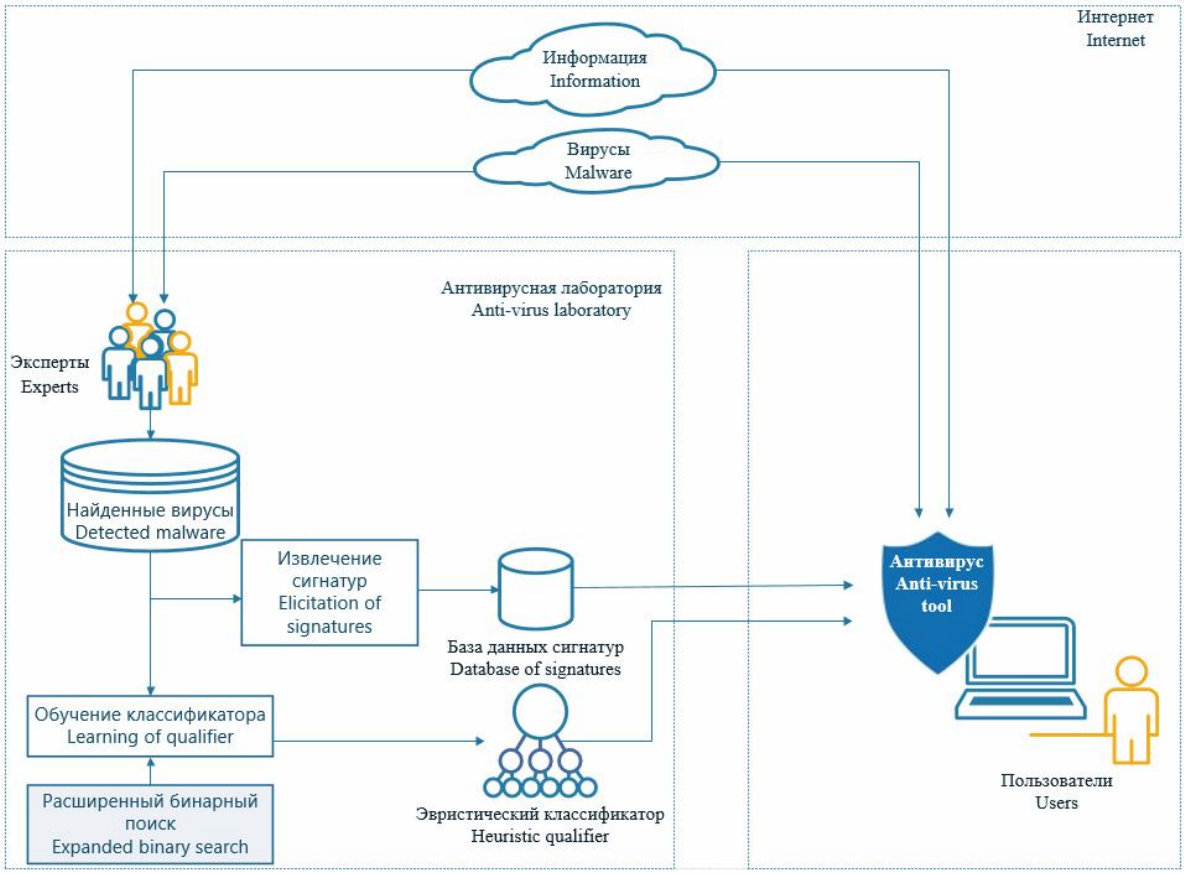
Azhmukhamedov Iskandar M., D.Sc. (Engineering), Associate Professor, Astrakhan State University, 20a Tatishchev St., Astrakhan, 414056, Russian Federation, e-mail: iskander_agm@mail.ru

Gurskaya Tatiana G., Ph.D. (Engineering), Associate Professor, Astrakhan State University, 20a Tatishchev St., Astrakhan, 414056, Russian Federation. e-mail: gurskai@mail.ru

The main means of anti-virus detection is the signature analysis. However, it isn't able to withstand "zero day" malwares, i.e. viruses which have not been studied by anti-virus experts yet. The heuristic analysis (HA) is applied to countermeasures to new viruses whose signatures haven't been included in antiviruses databases. Static HA is a specific instance of HA. Binary classification underlies HA and provides two stages: learning and detecting. At learning stage the features are taken from the known viruses on the basis of which the qualifier is formed. This qualifier can protect users from "zero day" malwares during the usage of the anti-virus packet installed on the user's computer. The most time-consuming step of qualificator learning is the stage of a features file selection and their insertion in the general list of features. These features characterize the file as a malware or well-behaved file. Several methods of the solution of specified tasks have been analyzed and their shortcomings for this stage have been revealed. The alternative method under the form of an algorithm of expanded binary search which transforms any list of features into the sorted sequence of unique elements has been suggested. Complexity of the offered algorithm has been appraised. The best and worst cases for this algorithm have been considered. The efficiency of the offered approach is confirmed as a result of its checking with the usage of computing experiments.

Keywords: anti-virus heuristic analysis, sorting, binary search, linear search, complexity of algorithms, machine learning, bitmaps, sorting by simple inserts, binary classification, learning qualifier

Graphical annotation (Графическая аннотация)



Введение. Развитие информационных технологий обеспечило пользователям широкие возможности оперативного доступа к информации. Однако это повлекло за собой и распространение разнообразного вредоносного программного обеспечения (ПО). Антивирусные лаборатории занимаются поиском вирусов, выделяют характеризующие их сигнатуры и добавляют их в базы данных (БД) антивирусного ПО (АВПО).

Обновления этих БД в АВПО, установленном на пользовательских ПЭВМ и серверах, чаще всего осуществляются в автоматическом режиме несколько раз в сутки (при условии, что ЭВМ подключены к Интернету). После установки обновлений пользователи получают защиту от выявленных экспертами новых вирусов. Однако те вирусы, сигнатуры которых еще не включены в БД, не обнаруживаются АВПО.

Существенно, что обновления БД для АВПО на пользовательских ЭВМ осуществляются с задержкой по времени, включающей в себя следующие компоненты: продолжительность выявления новых вирусов с момента их фактического попадания в сеть Интернет; время, требуемое на пересылку (поступление) «образцов вирусов» в антивирусные лаборатории – в т.ч. и в режиме автоматической пересылки подозрительных файлов с ПЭВМ пользователей; время, затрачиваемое в таких лабораториях на анализ поступившей информации и подготовку обновлений для БД АВПО (а иногда и для корректировки «ядра» АВПО); продолжительность ожидания пользователями момента очередного автоматического скачивания/установки обновленных БД АВПО на ПЭВМ/серверах.

Поэтому для обнаружения неизвестного ранее вредоносного ПО во всех современных АВПО, помимо сигнатурного анализа, дополнительно применяется эвристический анализ (ЭА). Он называется динамическим, если его проведение предусматривает запуск программы ЭА в виртуальном пространстве («песочнице»). Данный метод требует большого количество ресурсов ПК, поэтому его использование не всегда целесообразно. От этого недостатка свободен статический ЭА, не предусматривающий запуска программы ЭА в «песочнице». В ходе статического ЭА определяется, содержит ли в себе сканируемый файл признаки вирусов. Для этого решается задача бинарной классификации. Целью данной статьи было представление разработанного алгоритма, обеспечивающего повышение вычислительной эффективности выполняемых операций.

Общая характеристика проблематики исследований. Задача антивирусной классификации состоит из двух этапов: обучения классификатора и выполнения распознавания с целью оценки качества обучения [14]. Обучение осуществляется «с учителем» и в общем случае происходит следующим образом. «Учитель» каждому элементу обучающего множества (как правило, это исполняемые файлы и библиотеки) присваивает метку, отражающую, к какому классу относится каждый из файлов вредоносных или доброкачественных объектов. Из каждого объекта обучающего множества выделяются характеризующие его признаки, затем они добавляются в общий перечень признаков.

В качестве признаков файлов часто выступают n-граммы – сегменты последовательных n байт внутри исполняемого файла [8, 9, 11, 15, 16]. При этом каждая n-грамма рассматривается как признак. Всего в файле содержится $(L-(n-1))$ n-грамм, где L – общий размер файла в байтах. Обычно используются триграммы или 4-граммы. Общее число возможных 3-грамм составляет 256^3.

Рассмотрим пример представления файла в виде совокупности 3-грамм. Пусть некоторый файл был считан побайтово. При этом байты равны таким значениям: 1 54 6 78 0. В этом списке содержатся такие 3-граммы, как: (1 54 6), (54 6 78), (6 78 0) и т.д. Преобразуем 3-грамму (1 54 6) в формат натурального числа в десятичной системе счисления.

1. Каждый байт 3-граммы (1 54 6) переводим в двоичный вид, что дает (00000001 00110110 00000110). Важно, чтобы число вне зависимости от его величины было представлено восьмью разрядами.

2. Полученная битовая строка рассматривается как двоичная запись одного числа в виде 000000010011011000000110.

3. Это двоичное число переводится в десятичную систему счисления, что в нашем случае дает 79366.

Таким образом, побайтово считанный файл можно представить в виде списка n-грамм, а каждую n-грамму записать в виде десятичного числа.

После формирования общего перечня признаков необходимо из них выделить наиболее значимые (информативные), а избыточные и нерелевантные – отбросить [10, 17, 18]. Признак считается нерелевантными (недифференцирующим), если на его основе нельзя отличить объект одного класса от объекта другого класса. Например, пусть какая-либо n-грамма одинаково часто встречается как в доброкачественных, так и во вредоносных файлах. Тогда она не может быть включена в перечень информативных признаков, поскольку ее наличие в файле неизвестного типа не позволит уменьшить неопределенность отнесения данного файла к тому или иному классу.

На основе выделенных из файлов обучающего множества релевантных признаков в антивирусной лаборатории с использованием различных моделей («наивного» Байесовского классификатора; алгоритма J48; деревьев решений; «случайного леса»; градиентного бустинга) строится эвристический классификатор [8, 9, 11].

На этапе обнаружения из сканируемого (проверяемого) файла на пользовательской ЭВМ извлекаются его признаковые характеристики. Они сравниваются с характеристиками, выделенными в процессе обучения классификатора; определяется степень их соответствия признакам вредоносного ПО.

В случае если степень соответствия выше некоторого порогового значения, то проверяемый файл с определенной вероятностью признается (считается) вирусным. На этапе распознавания может быть выяснено, что сканируемый файл состоит на x % из n-грамм, свойственных вредоносным файлам, и на y % из n-грамм, свойственных доброкачественным файлам. При этом согласно настройкам «чувствительности» ЭА, определяющим пороговое значение, файл может быть признан (или не признан) вредоносным. Соответственно, чем чувствительней эвристика, тем больше вирусов будет обнаружено. Однако при этом увеличится и количество ложных срабатываний алгоритма на доброкачественных файлах.

Каждый файл обучающего множества содержит в себе большое количество признаков. Прежде чем будет принято решение о важности того или иного извлеченного признака, он должен быть добавлен в общий перечень признаков, который увеличивается после каждого рассмотренного при обучении объекта. Обучающее множество должно быть достаточно большим для обеспечения качества работы ЭА. Вследствие этого на обработку файлов обучающего множества тратится значительное количество вычислительных и временны**х** ресурсов. Таким образом, этап обучения классификатора является первостепенным (определяющим качество результатов классификации) и весьма затратным по времени. При этом этап отбора уникальных признаков файлов является одной из наиболее долгих операций [2]. Поскольку скорость обучения и переобучения эвристического классификатора крайне важна для своевременного реагирования на новые угрозы, то возникает задача сокращения времени отбора уникальных признаков файлов.

Постановка задачи. Рассмотрим следующую типовую ситуацию. Из анализируемого файла извлечен ряд признаков в виде n-грамм, некоторые из которых дублируются несколько раз [3, 7]. Необходимо удалить все повторяющиеся элементы списка, поскольку дубликаты не несут новой (дополнительной) информации. Выделенные таким образом оставшиеся (не повторяющиеся) признаки добавляются в общий перечень признаков. Хранить данные в общем перечне удобно в отсортированном виде, поскольку в этом случае процедура вставки (обновления) новых признаков будет осуществляться быстрее.

Исходя из этого, конкретную цель данной работы можно сформулировать следующим образом: разработать алгоритм, позволяющий за минимальное время излеченную из объекта совокупность признаков преобразовать в отсортированный перечень уникальных элементов.

Как было показано выше, n-грамма может быть представлена в виде десятичного числа. Поэтому далее в статье в качестве элементов обрабатываемого списка будут выступать (рассматриваться) натуральные числа в десятичной системе счисления. Однако необходимо отметить, что элементами списка могут быть любые объекты, для которых существует (может быть реализована) операция сравнения [1, 5, 6, 13].

Сравнительный анализ возможных алгоритмических решений для рассматриваемой задачи. Есть несколько способов решения задачи получения из произвольного (случайного) набора данных отсортированной последовательности уникальных (т.е. не повторяющихся) элементов [2].

Метод 1. Убрать из исходного набора данных все дублирующие элементы. Затем отсортировать все оставшиеся элементы в необходимом порядке.

Метод 2. Отсортировать исходный набор данных – в результате все «дубликаты» будут находиться на соседних позициях. После этого удалить все элементы, которые «равны» предыдущим (совпадают с ними).

Метод 3. Применить «метод битовых карт» для отбора уникальных элементов [7, 12]. Затем их упорядочить.

Рассмотрим каждый из этих методов подробнее.

1. Решение, основанное на удалении дубликатов и последующей сортировке (метод 1). Этот алгоритм является наиболее затратным по времени. Общая продолжительность выполнения алгоритма в этом случае складывается из времени удаления дубликатов и времени сортировки. Сложность линейного поиска одного элемента оценивается как $O(n)$. Для n элементов эта сложность равна $O(n^2)$. Сложность наиболее эффективных и часто применяемых на практике алгоритмов сортировок – $O(n*\log(n))$ [7]. Таким образом, общая сложность рассматриваемого алгоритма составляет $O(n^2 + n*\log(n)) = O(n(n + \log(n)))$.

2. Решение, основанное на сортировке и последующем удалении дубликатов (метод 2). В случае использования этого алгоритма основные затраты времени приходятся на сортировку. При этом для удаления повторяющихся элементов достаточно всего одного прохода по отсортированной последовательности. Общая сложность алгоритма будет составлять $O(n*\log(n) + n) = O(n(1 + \log(n)))$. Поэтому этот способ в вычислительном отношении более эффективен по сравнению с методом 1.

3. Решение, основанное на применении битовых карт (метод 3) [12]. Этот метод подходит лишь для элементов, значения которых могут быть представлены целыми числами в ограниченном диапазоне. Если известно максимально возможное значение элемента в списке (N_{max}), то создается битовая карта, содержащая N_{max} битов. Битовая карта представляет собой квадратную матрицу, содержащую $\sqrt{N_{\max}}$ строк/столбцов. Так, если известно, что на вход алгоритма поступают числа от 0 до 15 (всего 16 возможных значений), то создается матрица размером 4x4. Если N_{max} не является квадратом целого числа, то значение N_{max} становится равным ближайшему числу, которое больше него и равно квадрату целого числа. Например, если N_{max} равно 19, то вместо него берется число 25.

Затем путем перебора всех элементов списка устанавливается соответствующий бит в карте для каждого элемента. Число, которое необходимо занести в карту, представляется в двоичном виде. Полученная строка «разрезается» пополам. Вновь полученные числа переводятся в десятичную систему счисления и становятся координатами для битовой карты. Например, необходимо в битовую карту внести отметку о числе 12. В двоичной системе счисления 12 выглядит как «1100». «Разрезав» строку пополам получаем «11» и «00», что в десятичном виде соответствует «3» и «0». Получается, что отметка о том, что число 12 есть в списке, заносится в битовую карту в 3-ю строку в 0-ой столбец. Соответствующий бит меняет значение с «0» на «1». После этого для получения отсортированной последовательности элементы считываются из битовой карты в определенном порядке.

Описанный метод имеет ряд существенных недостатков. Во-первых, не всегда заранее известен диапазон входных данных. Во-вторых, если N_{max} велико, то может требоваться слишком много оперативной памяти ЭВМ. Кроме того, она может нерационально использоваться. Например, выделено памяти из расчета N_{max}, равного 1024, а при этом необходимо записать информацию о всего 10 числах. Как следствие, 90 % выделенной памяти не будет использовано. В-третьих, к отсортированному списку уникальных элементов невозможно обратиться в произвольный момент времени. Каждый раз, когда возникает необходимость обратиться к списку, его придется формировать заново. Поскольку в битовой карте могли произойти изменения.

Сложность метода 3 составляет $O(n)$, что крайне привлекательно с точки зрения производительности. Однако ограничение диапазона входных данных и другие описанные недостатки делают этот метод неприемлемым для решения поставленной ранее задачи. Кроме того, «простота восприятия» этого метода существенно ниже, чем у описанных ранее.

Метод решения, основанный на бинарном поиске. Как видно из описанных ранее методов 1 и 2, наиболее затратным по времени этапом является сортировка элементов. При большой размерности задачи время формирования отсортированного перечня уникальных элементов может стать недопустимо большим. Это время сокращается, если полностью отказаться от механизма сортировки в пользу бинарного поиска (БП). Однако БП можно применять только на отсортированной последовательности объектов, в то время как по условию рассматриваемой задачи на вход классифицирующего алгоритма подается случайный набор данных.

Поскольку непосредственно применить алгоритм БП оказывается невозможным, то предлагается модифицировать его, расширив перечень решаемых при этом задач. В классическом случае при БП ищется ответ на вопрос, содержится ли заданный элемент в отсортированном списке [4, 7]. Однако при этом, в случае если заданный элемент не найден, то можно в результате выполнения алгоритма дополнительно получать на выходе позицию. После вставки на нее «проверяемого» элемента, отсортированная последовательность таковой и останется. Назовем описанную модификацию алгоритма расширенным бинарным поиском (РБП) [2]. Необходимо отметить, что РБП может также рассматриваться и как модификация алгоритма сортировки простыми вставками [7], который часто применяется как часть других алгоритмов обработки списков.

Рассмотрим пример. Пусть имеется общий перечень n-грамм, который в десятичном представлении выглядит следующим образом {1; 3; 5; 7}. Из файла обучающего множества была извлечена очередная n-грамма, которой соответствует десятичное значение, равное 4. Необходимо узнать, имеется ли в последовательности n-грамма «4». РБП позволит не только определить то, что элемент «4» не входит в анализируемую последовательность, но также и вставить «4» на третью позицию (т.е. между 3 и 5). Причем после вставки последовательность элементов останется отсортированной по возрастанию.

Рассмотрим теперь, как РБП может быть использован для формирования отсортированной последовательности уникальных чисел.

Пусть имеется случайная последовательность, возможно, содержащая в себе повторяющиеся элементы. Будем «перекладывать» элементы в новую последовательность, которая изначально не содержит ни одного элемента, т.е. является «пустой». Берем первый элемент исходной (входной) последовательности и переносим его в искомую (новую, выходную) последовательность. Последовательность из одного элемента является отсортированной. В связи с этим по отношению к ней можно применить бинарный поиск.

Берем следующий элемент из исходной последовательности. Проверяем его наличие в выходной последовательности. Если он присутствует, то вставлять его повторно в выходную последовательность не требуется. Если он отсутствует, то РБП позволит определить на какую позицию его необходимо поставить в выходной последовательности так, чтобы она продолжала оставаться отсортированной. Таким образом, последовательно осуществляется «проверка – вставка» всех элементов исходной последовательности. В итоге, без реализации отдельного этапа сортировки получается отсортированная выходная последовательность, состоящая из уникальных элементов.

Описанный подход является достаточно простым для понимания и программирования. Однако если не применять специальные алгоритмические решения, то возникает необходимость использования «удвоенного» объема оперативной памяти для хранения «входной» и «выходной» последовательностей.

Сложность бинарного поиска одного элемента в выходной последовательности в этом случае составляет $O(\log(n))$ [7]. Поскольку поиск выполняется для каждого элемента этой последовательности, то итоговая сложность равна $O(n*\log(n))$, где n – количество элементов.

Существенным достоинством РБП является тот факт, что с его помощью можно осуществлять динамическое формирование отсортированного перечня уникальных элементов. Например, пусть на вход эвристического классификатора с некоторой периодичностью поступают данные. Эти данные необходимо хранить в отсортированном виде, без дубликатов. РБП позволяет, не дожидаясь, пока все данные будут собраны, формировать в режиме реального времени отсортированный список – т.е. последовательно корректировать его по мере поступления данных.

Таким образом, применение данного алгоритма позволяет в любой момент времени получить полный список признаков файлов в отсортированном виде без дубликатов. Ни один из рассмотренных выше алгоритмов-аналогов не может этого обеспечить.

Необходимость формирования отсортированного списка по мере поступления данных возникает при отслеживании эффективности процесса обучения ЭА, т.е. определения того, как меняется список признаков после обработки каждого файла обучающего множества; какой при этом имеет место прирост информации (на сколько уменьшилась энтропия) и др.

«Лучшим» случаем при оценке трудоемкости для РБП будет являться ситуация, когда каждый элемент из входной последовательности будет либо совпадать с первым сравниваемым (центральным, стоящим в середине выходного списка) элементом, либо середина выходного списка является местом для вставки. В случае если количество элементов четное, то центральным будем считать элемент, стоящий справа от середины интервала, в рамках которого проходил поиск. Пример работы РБП в «лучшем» случае представлен в таблице 1. При этом для простоты в качестве элементов списка используется ограниченное количество натуральных, не повторяющихся чисел. Жирным шрифтом выделены элементы, обрабатываемые (переносимые в выходную последовательность) на текущем шаге алгоритма.

Таблица 1 – **Пример работы РБП в «лучшем» случае**

№ итерации (шага алгоритма)	Входная последовательность	Выходная последовательность
1.	**1** 10 2 9 3 8 4 7 5 6	**1**
2.	**10** 2 9 3 8 4 7 5 6	1 | **10**
3.	**2** 9 3 8 4 7 5 6	1 **2** | 10
4.	**9** 3 8 4 7 5 6	1 2 | **9** 10
5.	**3** 8 4 7 5 6	1 2 **3** | 9 10
6.	**8** 4 7 5 6	1 2 3 | **8** 9 10
7.	**4** 7 5 6	1 2 3 **4** | 8 9 10
8.	**7** 5 6	1 2 3 4 | **7** 8 9 10
9.	**5** 6	1 2 3 4 **5** | 7 8 9 10
10.	**6**	1 2 3 4 5 | **6** 7 8 9 10

На первой итерации из входной последовательности в выходную переносится первое значение, т.е. «1». На второй итерации, текущий элемент, равный «10», сравнивается с «1» в выходной последовательности и переносится справа от «1». На третьей итерации «2» сравнивается с «10». Так как $2 < 10$, то «2» сравнивается еще и с «1». Поскольку $2 > 1$, то «2» переносится на позицию после «1». Аналогично выполняются последующие шаги.

«Худшим» по трудоемкости случаем для РБП будет являться ситуация, когда для вставки элемента в выходную последовательность потребуется максимальное число сравнений. Так, например, для вставки числа 2 в последовательность {1; 3; 4; 5; 6; 7; 8; 9; 10} потребуется четыре сравнения. Сначала «2» сравнивается с «6», стоящей в центре списка на пятой позиции. Так как $2 < 6$, то дальше рассматривается только левая половина списка {1; 3; 4; 5}. Далее «2» сравнивается «4». При этом «2» опять оказывается меньше. Поэтому дальнейший поиск позиции будет происходить для левой половины {1; 3}. Величина «2» сравнивается поочередно с «3» и «1». На основании сравнений делается вывод, что «2» в выходной последовательности нет. Поэтому и вставить ее необходимо на вторую позицию, т.е. между «1» и «3».

Как видно из приведенного описания РБП, количество итераций при формировании отсортированной последовательности уникальных элементов зависит от количества сравнений в рамках проведения бинарного поиска (эти сравнения необходимы для проверки наличия элемента в выходной последовательности).

Экспериментальная проверка вычислительной эффективности предложенного алгоритма. Сравним по продолжительности выполнения методов «1» и «2» с РБП в ситуации, когда входная последовательность чисел случайна (табл. 2), а сами числа натуральные и располагаются в диапазоне от «0» до «10000». Доли повторяющихся чисел в последовательностях составляют 50 %.

Для проведения вычислительных экспериментов на языке C^{++} был разработан программный продукт в среде Microsoft Visual Studio 2008. Алгоритм РБП был реализован авторами самостоятельно. Программные реализации других способов сортировки были взяты из встроенной библиотеки среды разработки. Отсчет времени производился с помощью разницы значений, даваемых функцией языка C^{++} «*clock*» до начала и после окончания вычислений в каждом из экспериментов. При этом использовалась ПЭВМ со следующими характеристиками: процессор Intel(R) Core(TM) i3 CPU M330 @2.13GHz 2.13GHz; ОЗУ 3,00 Гб.

Как видно из таблицы 2, при любом объеме входных данных РБП является более эффективным в вычислительном отношении (более быстрым) по сравнению с другими методами.

Таблица 2 – **Результаты сравнения вычислительной эффективности различных методов (длительности работы алгоритмов даны в секундах)**

Количество элементов во входной последовательности *n (шт.)*	**1-ый метод: удаление и сортировка, *с***	**2-ой метод: сортировка и удаление, *с***	**РБП, *с***
10 000	0.929	0.394	0.030
20 000	1.887	0.791	0.049
30 000	3.091	1.247	0.064
40 000	4.016	1.602	0.103
50 000	5.248	1.962	0.105

Для подтверждения соответствия реальной вычислительной сложности РБП заявленной теоретической оценке $O(n*\log(n))$ был проведен дополнительный вычислительный эксперимент (на той же ПЭВМ, что использовалась для первого эксперимента). Результаты второго эксперимента представлены в таблице 3.

Таблица 3 – **Результаты экспериментальной проверки вычислительной сложности алгоритма РБП**

Количество элементов в последовательности *n (шт.)*	***n**log(*n*)**	***t*, *с***
10000	92103.40	0.1196
20000	198069.75	0.2861
30000	309268.58	0.4927
40000	423865.38	0.7125
50000	540988.91	0.9507
60000	660125.99	1.1904
70000	780937.53	1.4351
80000	903182.55	1.6877
90000	1026680.85	1.9517
100000	1151292.55	2.2094

Для сравнения фактической зависимости времени работы РБП с заявленной сложностью $O(n*\log(n))$ были рассчитаны значения $o = n*\log(n)$ и построен график $t(o)$ (рис.).

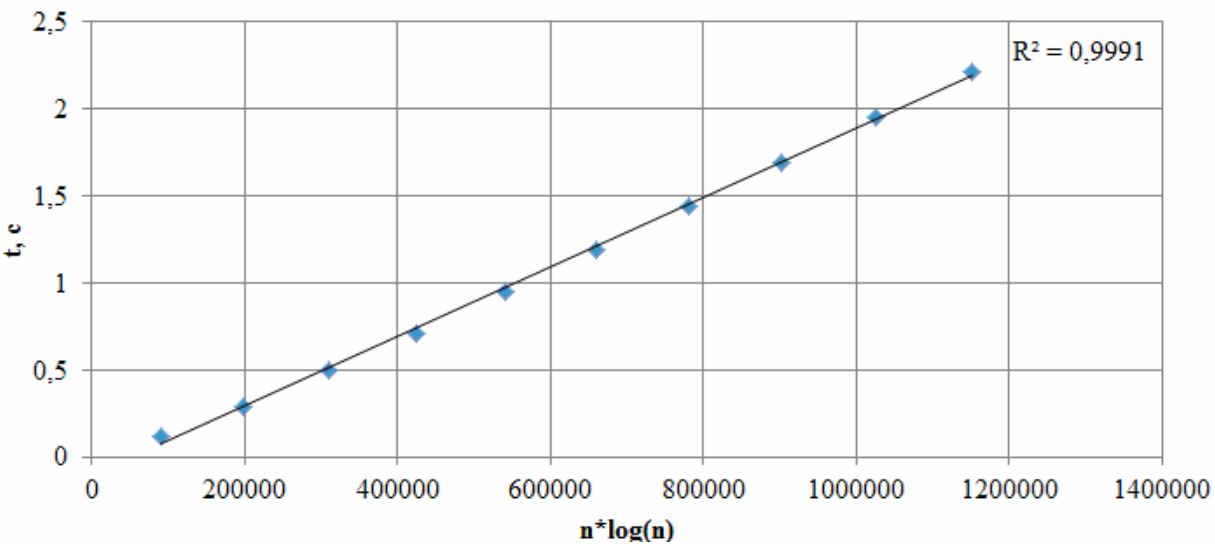


Рисунок – График t(o)

Величина достоверности линейной аппроксимации (R^2) составляет 0,9991, что фактически подтверждает заявленную сложность алгоритма РБП.

Выводы. Этап обучения в задачах классификации является основополагающим и крайне затратным с точки зрения временных ресурсов. Для ускорения вычислений на данном этапе предложен алгоритм РБП, обладающий значительным преимуществом перед другими алгоритмами в отношении вычислительной эффективности. Это было подтверждено в ходе экспериментального сравнения РБП с другими методами. Теоретически оцененная сложность РБП (зависимость времени выполнения от количества входных элементов) в виде O(n*log(n)) была экспериментально подтверждена на наборах случайных данных различного объема.

Список литературы

1. Ажмухамедов И. М. Методика оценки уровня безопасности информационных активов на основе нечетких продукционных правил / И. М. Ажмухамедов, О. М. Князева // Проблемы информационной безопасности. Компьютерные системы. – 2015. – № 1. – С. 7–16.

2. Астахова Н. Д. Сравнительный анализ вариантов оптимизации при разработке моделей прогнозирования на основе строго бинарных деревьев / Н. Д. Астахова, Л. А. Демидова // Прикаспийский журнал: управление и высокие технологии. – 2016. – № 2. – С. 9–25.

3. Введение в анализ сложности алгоритмов // Хабрахабр: техноблог. 08.10.2013. – Режим доступа: http://habrahabr.ru/post/195996/ (дата обращения: 21.12.2016), свободный. – Заглавие с экрана. – Яз. рус.

4. Демина Р. Ю. Использование бинарного поиска для формирования отсортированного перечня уникальных элементов / Р. Ю. Демина, И. М. Ажмухамедов // Математические методы в технике и технологиях – ММТТ-27. – Тамбов : Тамбовский государственный технический университет, 2014. – С. 124–126.

5. Дюдикова Е. И. Анализ и управление рисками использования информационных технологий при работе с наличными, безналичными и электронными деньгами / Е. И. Дюдикова, Ю. М. Брумштейн, Н. Ю. Танющева, Р. Ю. Демина, Е. Ю. Васьковский, А. Б. Кузьмина, И. А. Дюдиков // Прикаспийский журнал: управление и высокие технологии. – 2016. – № 1. – С. 161–175.

6. Жилин Л. Е. Анализ номенклатуры программных средств массового использования, применяемых в российских вузах (на примере Астраханского государственного университета) / Л. Е. Жилин, А. Н. Горбачева, Ю. М. Брумштейн, Е. Ю. Васьковский // Прикаспийский журнал: управление и высокие технологии. – 2015. – № 2. – С. 20–37.

7. Кнут Д. Э. Искусство программирования. Сортировка и поиск : пер. с англ. / пер. : В. Тертышный, И. Красиков. – Москва : Вильямс, 2012. – 824 с.

8. Курочкин А. Г. Использование гибридных нейросетевых моделей для многоагентных систем классификации в гетерогенном пространстве информативных признаков / А. Г. Курочкин, В. В. Жилин, С. Е. Суржикова, С. А. Филист // Прикаспийский журнал: управление и высокие технологии. – 2015. – № 3. – С. 85–95.

9. Нгуен Туан Ань. Разработка метода про-активного обнаружения мошенничества потребителей услуг телекоммуникационной компании / Нгуен Туан Ань, М. В. Щербаков, Чан Ван Фу, А. Г. Кравец // Прикаспийский журнал: управление и высокие технологии. – 2016. – № 4. – С. 43–52.

10. Потапов А. С. Распознавание образов и машинное восприятие / А. С. Потапов. – Санкт-Петербург : Политехника, 2007. – 552 с.

11. Путин Е. О. Классификатор для статического обнаружения компьютерных вирусов, основанный на машинном обучении / Е. О. Путин, А. В. Тимофеев // Information Technologies & Knowledge. – 2014. – № 2. – С. 103–112.

12. Савчук И. Г. Оценка количества уникальных элементов в большом списке / И. Г. Савчук // Blogerator.ru: Авторский техноблог. – Режим доступа: http://blogerator.ru/page/ocenka-kolichestva-unikalnyh-elementov-v-bolshom-spiske-element-spiskaalgoritm-sortirovki (дата обращения: 21.12.2016), свободный. – Заглавие с экрана. – Яз. рус.

13. Финогеев А. Г. Анализ и классификация атак через беспроводные сенсорные сети в SCADA системах / А. Г. Финогеев, И. С. Нефедова, Е. А. Финогеев, Куанг Винь Тхай, П. В. Ботвинкин // Прикаспийский журнал: управление и высокие технологии. – 2014. – № 1. – С. 12–23.

14. Флах П. Машинное обучение. Наука и искусство построения алгоритмов, которые извлекают знания из данных / П. Флах. – Москва : ДМК Пресс, 2015. – 400 с.

15. Axbou-Assaleh T. N-gram-based Detection of New Malicious Code Privacy and Security Laboratory / T. Axbou-Assaleh, N. Cercone, V. Keˇselj, R. Sweidan // Faculty of Computer Science, Dalhousie University. – 2004. – № 4. – Режим доступа: vxheaven.org/lib/pdf/N-gram-based%20Detection%20of% 20New% 20Malicious%20Code.pdf, свободный. – Заглавие с экрана. – Яз. рус.

16. Harrington P. Machine Learning in Action / P. Harrington. – Manning Publications Co, 2012. – 382 p.

17. Marsland S. Machine Learning: An Algorithmic Perspective / S. Marsland. – 2nd ed. – Chapman and Hall/CRC, 2014. – 457 p.

18. Tahan G. Mal-ID: Automatic Malware Detection Using Common Segment Analysis and Meta-Features / G. Tahan, L. Rokach, Y. Shahar // Journal of Machine Learning Research. – 2012. – № 13. – P. 949–979.

References

1. Azhmukhamedov I. M., Knyazeva O. M. Metodika otsenki urovnya bezopasnosti informatsionnykh aktivov na osnove nechetkikh produktsionnykh pravil [Assessments of the level of security of information asset methodology based on fuzzy production rules]. Problemy informatsionnoy bezopasnosti. Kompyuternye sistemy [Information Security Problems. Computer Systems], 2015, no. 1, pp. 7–16.

2. Astakhova N. D., Demidova L. A. Sravnitelnyy analiz variantov optimizatsii pri razrabotke modeley prognozirovaniya na osnove strogo binarnykh derevev [Comparative analysis of the optimization variants for the development of the forecasting models on the base of the strictly binary trees]. *Prikaspiyskiy zhurnal: upravlenie i vysokie tekhnologii* [Caspian Journal: Control and High Technologies]. 2016, no. 2, pp. 9–25.

3. Vvedenie v analiz slozhnosti algoritmov [Introduction to the analysis of complexity of algorithms]. *Khabrakhabr: tekhnicheskiy blog* [Habrahabr: technical blog]. Available at: http://habrahabr.ru/post/195996/ (accessed 21.12. 2016).

4. Demina R. Yu., Azhmukhamedov I. M. Ispolzovanie binarnogo poiska dlya formirovaniya otsortirovannogo perechnya unikalnykh elementov [Use of binary search for formation of the sorted list of unique elements]. *Matematicheskie metody v tekhnike i tekhnologiyah MMTT-27* [Mathematical Metods in Technique and Technologies – MMTT-27], Tambov, Tambov State Tekhnical University Publ. House, 2014, pp. 124–126.

5. Dyudikova Ye. I., Brumshteyn Yu. M., Tanyushcheva N. Yu., Demina R. Yu., Vaskovskiy Ye. Yu., Kuzmina A. B., Dyudikov I. A. Analiz i upravlenie riskami ispolzovaniya informatsionnykh tekhnologiy pri rabote s nalichnymi, beznalichnymi i elektronnymi dengami [The analysis and risk management of information technologies usage during the work with cash, non-cash and electronic money]. *Prikaspiyskiy zhurnal: upravlenie i vysokie tekhnologii* [Caspian Journal: Control and High Technologies], 2016, no. 1, pp. 161–175.

6. Zhilin L. Ye., Gorbacheva A. N., Brumshteyn Yu. M., Vasilkovskiy Ye. Yu. Analiz nomenklatury programmnykh sredstv massovogo ispolzovaniya, primenyaemykh v rossiyskikh vuzakh (na primere Astrakhanskogo gosudarstvennogo universiteta) [Nomenclature analysis of mass usage software, applied in Russian Universities (on the example of Astrakhan State University)]. *Prikaspiyskiy zhurnal: upravlenie i vysokie tekhnologii* [Caspian Journal: Control and High Technologies], 2015, no. 2, pp. 20–37.

7. Knut D. E. *Iskusstvo programmirovaniya. Sortirovka i poisk* [Art of Computer Programming. Sorting and Searching], Moscow, Vilyams Publ., 2012. 824 p.

8. Kurochkin A. G., Zhilin V. V., Surzhikova S. Ye., Filist S. A. Ispolzovanie gibridnykh neykrosetevyh modeley dlya mnogoagentnykh sistem klassifikatsii v geterogennom prostranstve informativnykh priznakov [Use of hybrid neural network models for mnogoagentny systems of classification in heterogeneous space of informative signs]. *Prikaspiyskiy zhurnal: upravlenie i vysokie tekhnologii* [Caspian Journal: Control and High Technologies], 2015, no. 3, pp. 85–95.

9. Nguen Tuan An, Shcherbakov M. V., Chan Van Fu, Kravets A. G. Razrabotka metoda proaktivnogo obnaruzheniya moshennichestva potrebiteley uslug telekommunikatsionnoy kompanii [Development of a method of pro-active detection of a fraud of consumers of services of the telecommunication company]. *Prikaspiyskiy zhurnal: upravlenie i vysokie tekhnologii* [Caspian Journal: Control and High Technologies], 2016, no. 4, pp. 43–52.

10. Potapov A. S. *Raspoznavanie obrazov i mashinnoe vospriyatie* [Recognition of images and machine perception], Saint Petersburg, Politekhnika Publ., 2007. 552 p.

11. Putin Ye. O., Timofeev A. V. Klassifikator dlya staticheskogo obnaruzheniya kompyuternykh virusov, osnovannyy na mashinnom obuchenii [Classifier for static detection of computer viruses, based on machine learning]. *Information Technologies & Knowledge*, 2014, no. 2, pp. 103–112.

12. Savchuk I. G. Otsenka kolichestva unikalnykh elementov v bolshom spiske [Assessment of quantity of unique elements in the big list]. *Blogerator.ru: Avtorskiy tekhnicheskiy blog* [Blogerator.ru: Authoring technical blog]. Available at: http://blogerator.ru/page/ocenka-kolichestva-unikalnyh-elementov-v-bolshom-spiske-element-spiskaalgoritm-sortirovki (accessed 21.12.2016).

13. Finogeev A. G., Nefedova I. S., Finogeev Ye. A., Kuang Vin Thay, Botvinkin P. V. Analiz i klassifikatsiya atak cherez besprovodnye sensornye seti v SCADA-sistemakh [Analysis and classification of attacks via wireless sensor networks in SCADA systems]. *Prikaspiyskiy zhurnal: upravlenie i vysokie tekhnologii* [Caspian Journal: Control and High Technologies], 2014, no. 1, pp. 12–23.

14. Flach P. *Mashinnoe obuchenie. Nauka i iskusstvo postroeniya algoritmov, kotorye izvlekayut znaniya iz dannykh* [Machine Learning. The art and Since of Algorithms that Make Sense of Data], Moscow, DMK Press Publ., 2015. 400 p.

15. Axbou-Assaleh T., Cercone N., Keˇselj V., Sweidan R. N-gram-based Detection of New Malicious Code Privacy and Security Laboratory. *Faculty of Computer Science, Dalhousie University*, 2004, no. 4. Available at: vxheaven.org/lib/pdf/N-gram-based%20Detection%20of%20New% 20Malicious%20Code.pdf.

16. Harrington P. *Machine Learning in Action*, Manning Publications Co Publ., 2012. 382 p.

17. Marsland S. *Machine Learning: An Algorithmic Perspective*. 2nd ed. Chapman and Hall/CRC, 2014. 457 p.

18. Tahan G., Rokach L., Shahar Y. Mal-ID: Automatic Malware Detection Using Common Segment Analysis and Meta-Features. *Journal of Machine Learning Research*, 2012, no. 13, pp. 949–979.