

**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук

УТВЕРЖДАЮ
Академический руководитель
образовательной программы
«Науки о данных»,

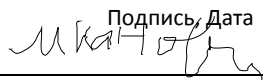

_____ С.О. Кузнецов
«___» _____ 2017 г.

Выпускная квалификационная работа

на тему: Бинарная классификация как вспомогательный инструмент для решения SAT задач

тема на английском языке: Binary Classification as a Complementary Tool for Solving SAT Problems

по направлению подготовки 01.04.02 «Науки о данных»

<p>Научный руководитель</p> <p>Профессор НИУ ВШЭ</p> <hr/> <p>Должность, место работы</p> <p>д. ф.-м. н.</p> <hr/> <p>ученая степень, ученое звание</p> <p>М.И. Канович</p> <hr/> <p>И.О. Фамилия</p> <p>Оценка 10</p> <hr/> <p>Подпись, Дата</p> <p> 25.05.2017</p>	<p>Выполнил</p> <p>студент группы м15НоД_ИССА</p> <p>2 курса магистратуры</p> <p>образовательной программы «Науки о данных»</p> <p>В.М. Мосин</p> <hr/> <p>И.О. Фамилия</p> <p> 22.05.2017</p> <hr/> <p>Подпись, Дата</p>
--	---

Москва 2017

Оглавление

Введение.....	3
1.1 Общий обзор проблемы.....	3
1.2 Актуальность задачи	5
1.3 Структура работы	6
Обзор литературы и существующих подходов	7
2.1 Проблема выполнимости булевых формул	7
2.1.1 SAT в стандартном формате.....	7
2.1.2 Фазовый переход.....	8
2.2 Алгоритмы решения SAT	9
2.2.1 Алгоритм DPLL.....	10
2.2.2 Пример работы алгоритма DPLL.....	11
2.3 Машинное обучение и SAT	12
2.3.1 Значимые признаки SAT формул.....	13
Идея и план разработки.....	15
3.1 Идея применения классификатора	15
3.2 Алгоритм выбора значения переменной ветвления	16
3.3 Общая схема внедрения машинного обучения	18
3.3.1 Построение обучающей выборки	19
3.3.2 Обучение классификаторов	19
Реализация	20
4.1 Техническая спецификация.....	20
4.2 Описание классов.....	20
4.2.1 Класс CNF	20
4.2.2 Класс DPLL.....	21
4.3 Описание вспомогательных функций	22
Экспериментальные результаты и анализ	23
5.1 Машинное обучение.....	23
5.2 Тестирование DPLL алгоритма	27
Заключение.....	33
6.1 Обзор проделанной работы.....	33
6.2 Программный код проекта	34
6.3 Направления дальнейших исследований.....	34

Глава 1

Введение

SAT алгоритм при выполнении исследует очень большое пространство всевозможных решений при поиске подходящего. В данной работе предлагается на каждом шаге работы алгоритма использовать предварительную бинарную классификацию SAT формул для того, чтобы максимально сократить это пространство поиска. Однако, выполнение дополнительной процедуры потребует времени, и поэтому увеличение общей производительности алгоритма не гарантировано. Целью данной работы является исследование целесообразности применения бинарного классификатора непосредственно в одном из существующих полных алгоритмов определения выполнимости булевых формул – DPLL, а также практическая реализация предложенной в работе методики на языке программирования. В данную задачу входит составление общей схемы внедрения бинарного классификатора в алгоритм, построение обучающей выборки, тренировка различных бинарных классификаторов, сравнение результатов их обучения и анализ производительности модифицированного алгоритма с применением наилучшего бинарного классификатора в сравнении со стандартным вариантом алгоритма.

1.1 Общий обзор проблемы

Проблема выполнимости булевых формул (англ. SAT – satisfiability problem) – это каноническая задача в области компьютерных наук, имеющая как практическую, так и теоретическую значимость. Впервые NP-полнота была доказана именно для этой задачи [4]. Такие NP-задачи как Задача коммивояжёра (Travelling salesman problem), Раскраска графов (Graph coloring) и другие сводятся к SAT за полиномиальное время. Соответственно, способность быстро решать SAT также дает возможность решать быстро все другие NP-задачи. Помимо представления теоретического интереса, SAT

проблема возникает и при решении практических задач из различных областей. К таким задачам относятся, например, верификация программного и аппаратного обеспечения [3], автоматическое доказательство теорем, проектирование цифровых схем, искусственный интеллект и планирование, а также многие другие. В связи с этим задача выполнимости булевых формул привлекает особый интерес со стороны научной общественности, а также проводятся ежегодные соревнования по решению SAT задач [15], на которых придумываются новые алгоритмы и улучшаются уже существующие.

В общем случае проблема выполнимости булевых формул формулируется следующим образом: для данной булевой формулы определить, является ли она выполнимой, то есть существует ли такой набор значений логических переменных, входящих в формулу, в результате присвоения которого данная булевая формула становится истинной. Например, следующая булева формула является выполнимой:

$$F = (x_1 \vee x_3 \vee \neg x_4) \wedge (x_4) \wedge (x_2 \vee \neg x_3).$$

Она принимает истинное значение на следующем наборе значений логических переменных:

$$x_1 = 1, x_2 = 0, x_3 = 0, x_4 = 1.$$

На данный момент существует большое количество алгоритмов решения SAT. Обычно на практике применяются полные алгоритмы, основанные на систематическом поиске, и неполные алгоритмы, в основном базирующиеся на локальном поиске решения. Первая группа алгоритмов имеет экспоненциальную сложность в наихудшем случае, так как SAT является NP-полной задачей. В то же время алгоритмы из второй группы, хоть и являются масштабируемыми, однако они не способны доказать невыполнимость булевой формулы.

Активное развитие области машинного обучения поспособствовало началу исследования применения данных технологий в решении SAT. Уже

проведена работа по извлечению значимых признаков из булевых формул. На данный момент существуют исследования, в которых эти признаки применяются для обучения моделей с целью последующего их применения для определения наилучшего алгоритма SAT из портфолио алгоритмов. С другой стороны, SAT может рассматриваться как задача бинарной классификации, то есть определения принадлежности конкретной булевой формулы к классу выполнимых или невыполнимых формул. Такой подход позволяет с некоторой точностью заранее судить о выполнимости булевой формулы.

1.2 Актуальность задачи

Полученные практические результаты по извлечению значимых признаков булевых формул успешно применяются для предсказания времени работы алгоритмов. Также применение бинарных классификаторов для предварительного определения выполнимости булевых формул является полезным на практике для получения ответа с некоторой долей вероятности. Однако пока не исследована возможность применения машинного обучения непосредственно в алгоритмах SAT.

В данной работе представлен подход, в котором бинарный классификатор используется для принятия решения о присвоении значений логических переменных при поиске решения прямо во время работы SAT алгоритма. Результаты исследования прироста производительности этого подхода по сравнению со стандартным алгоритмом свидетельствуют о его практической значимости, а также открывают возможности для дальнейшего углубленного изучения этой проблемы.

Реализация предложенного в работе подхода в программном коде является дополнительным артефактом данного проекта, обеспечивающим воспроизводимость полученных результатов для применения в последующих исследованиях.

1.3 Структура работы

Глава 2 (Обзор литературы и существующих подходов) – содержит детальное рассмотрение статей как по классической тематике SAT, так и по современным достижениям в области применения машинного обучения для решения задачи определения выполнимости булевой формулы. В этой части приводится описание алгоритма DPLL и существующих значимых признаков булевых формул.

Глава 3 (Идея и план разработки) – в этой части читателю представляется общая схема внедрения бинарного классификатора в работу SAT алгоритма, подробно описывается алгоритм присвоения значения логической переменной и его параметры, приводится обоснование его применения.

Глава 4 (Реализация) – данная глава описывает основные моменты процесса разработки алгоритмов на языки программирования, построения пространства признаков и обучения классификаторов, а также их реализация в коде. Здесь также определены классы и функции, используемые в работе.

Глава 5 (Экспериментальные результаты и анализ) – здесь приводятся описание построения экспериментальной части и основные результаты, полученные в ходе работы, проводится их анализ и делается вывод об условиях применения бинарного классификатора в алгоритме DPLL.

Глава 6 (Заключение) – заключительная глава суммирует главные аспекты работы и дает обзор потенциальных дальнейших исследований по данной теме.

Глава 2

Обзор литературы и существующих подходов

2.1 Проблема выполнимости булевых формул

Булева формула – это выражение, состоящее из логических переменных, связанных между собой базовыми логическими операторами: конъюнкцией, дизъюнкцией и отрицанием («И», «ИЛИ» и «НЕ»).

Проблема выполнимости булевых формул определяется следующим образом: дана булева формула $\varphi = f(x_1, \dots, x_n)$ на наборе логических переменных x_1, \dots, x_n ; определить, существует ли такое присвоение значений логических переменных, таких что формула φ становится истинной.

2.1.1 SAT в стандартном формате

Чаще всего булевы формулы в SAT задачах представлены в стандартном формате, называемом конъюнктивной нормальной формой (КНФ) (англ. CNF – conjunctive normal form). В этом случае булева формула записывается в виде конъюнкции выражений в скобках (далее просто «скобках»), где каждая скобка состоит из дизъюнкции логических переменных или их отрицаний (далее «литералов»). Например, SAT формула, представленная во введении этой работы, записана в CNF формате:

$$F = (x_1 \vee x_3 \vee \neg x_4) \wedge (x_4) \wedge (x_2 \vee \neg x_3).$$

Эта формула состоит из трех скобок: первая – дизъюнкция литералов x_1, x_3 и $\neg x_4$; вторая – состоит из одного литерала x_4 ; третья – дизъюнкция литералов x_2 и $\neg x_3$.

Существует также понятие как k -SAT формула. Это частный случай булевой формулы, записанной в CNF формате, когда каждая скобка формулы содержит не более k литералов. SAT задача является NP-полной, если $k \geq 3$. Любая k -SAT формула, где $k \geq 3$, может быть сведена к 3-SAT формуле за

полиномиальное время относительно размера формулы. Поэтому в данной работе в экспериментальной части будут использоваться именно 3-SAT формулы. К тому же этот вид формул является очень популярным среди организаторов соревнований по SAT задачам, так что экспериментальные данные можно легко заимствовать из существующих источников.

2.1.2 Фазовый переход

Одним из важных феноменов, исследуемых в SAT задачах, является понятие фазового перехода. На первый взгляд может показаться, что вероятность того, что SAT формула в CNF формате является выполнимой постепенно уменьшается от 1 до 0 с увеличением числа скобок в формуле. Однако, на самом деле переход из области высоких вероятностей в область низких происходит довольно резко и зависит от отношения числа скобок к количеству переменных в формуле. В работе [19], например, было исследовано, что для 3-SAT формул, сгенерированных случайным образом, такой переход соответствует значению $M/N = 4.26$, где M – это количество скобок в формуле, N – количество переменных. Этот феномен называется фазовым переходом и проиллюстрирован на Рисунке 1. На нем видно, как растет доля невыполнимых формул среди всех сгенерированных с увеличением значения M/N . Стоит также заметить, что фазовый переход сглаживается с увеличением числа переменных в формуле.

SAT формулы из области фазового перехода являются самыми сложными для решения, так как вероятности выполнимости формул, взятых из этой области далеки от значений 0 и 1. Более того, в работе [16] утверждается, что SAT формулы из области фазового перехода хотя и являются наиболее сложными для бинарной классификации, однако, все же возможно получить достаточно высокую точность, достигая даже 70% в некоторых случаях. В экспериментальной части данной работы используются формулы именно из области фазового перехода, потому что они хорошо подходят для определения пригодности как отдельно взятых моделей

классификации, так и всего подхода внедрения машинного обучения в SAT алгоритмы в целом.

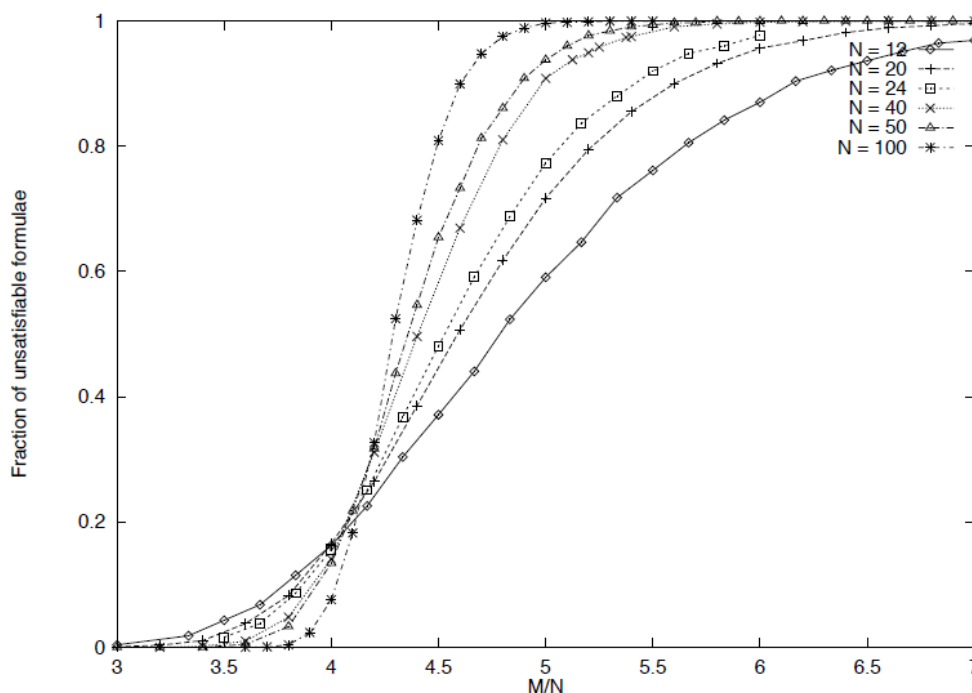


Рисунок 1. Фазовый переход

2.2 Алгоритмы решения SAT

Алгоритмы решения SAT делятся на две большие группы: полные и неполные методы. Полные методы решения SAT задач либо находят решение, удовлетворяющее условию выполнимости, либо доказывают, что формула невыполнима. Такие методы основываются на систематическом поиске в пространстве решений. Неполные SAT алгоритмы не гарантируют выдачу решения и обычно выполняются в течение предустановленного количества итераций, по истечению которого они могут найти, а могут и не найти конечного решения. Такие методы, в отличие от полных методов, основывающихся на систематическом поиске по всему пространству решений, базируются на стохастическом локальном поиске, включая использование генетических алгоритмов для решения SAT [18]. В некоторых случаях неполные алгоритмы превосходят полные методы по производительности.

В данной работе исследуется применение классификаторов для полного SAT алгоритма – DPLL.

2.2.1 Алгоритм DPLL

Название алгоритма DPLL (Davis-Putnam-Logemann-Loveland) является сокращением от фамилий исследователей, разработавших его. Работа данного алгоритма заключается в поиске по дереву решений и хорошо описана в статье [1]. Основной идеей алгоритма DPLL является то, что при таком поиске не рассматривается все пространство решений, а только те решения, которые не нарушают выполнимость отдельных скобок. Алгоритм 1 содержит псевдокод DPLL алгоритма, принимающего на вход булеву формулу F в формате CNF.

Алгоритм 1. DPLL(F)

```
 $F = \text{UnitPropagate}(F)$ 
if  $F$  contains the empty clause then return UNSAT
if  $F$  has no clauses left then return SAT
 $l = \text{a literal from } F$  // шаг ветвления
if DPLL( $F|_l$ ) = SAT then return SAT
if DPLL( $F|_{\neg l}$ ) = SAT then return SAT
return UNSAT

sub UnitPropagate( $F$ ):
    while  $F$  contains no empty clause but has a unit clause  $x$  do
         $F = F|_x$ 
    return  $F$ 
```

DPLL – рекурсивный алгоритм. Идея его работы состоит в выборе переменной l из формулы F и дальнейшем рекурсивном поиске для формулы $F|_l$ (зафиксировав $l = 1$) и для формулы $F|_{\neg l}$ (зафиксировав $l = 0$). Шаг, на котором выбирается переменная l , называется шагом ветвления, а сама переменная l – переменной ветвления. Для увеличения производительности алгоритм DPLL использует вспомогательную функцию UnitPropagate, которая находит все скобки, содержащие только один литерал на данном этапе работы алгоритма, фиксирует значение 1 для таких литералов во всей формуле и возвращает получившуюся формулу. Когда встречается формула, содержащая

пустую скобку, это означает, что данная ветвь дерева решений не подходит, возникает так называемый конфликт, и алгоритм должен делать шаг назад – выход из рекурсивного вызова функции.

Важным моментом в работе алгоритма является выбор переменной ветвления. Существуют различные критерии выбора переменной ветвления, которые существенным образом влияют на производительность алгоритма. В статье [17] приведено описание обширного исследования по сравнению различных критериев выбора переменной ветвления. В данной же работе алгоритм DPLL будет реализован с тремя базовыми критериями ветвления. Первый критерий – выбор переменной ветвления случайным образом (RANDOM). Согласно второму критерию (MAXO – maximum number of occurrences) выбирается переменная по максимальному количеству вхождений в формулу. Третий критерий аналогичен второму, но при нем переменная ветвления выбирается по максимальному вхождению в скобки минимального размера (MOMS – maximum number of occurrences in minimum size clauses).

Следующий раздел демонстрирует пример работы DPLL алгоритма.

2.2.2 Пример работы алгоритма DPLL

В примере на Рисунке 2 используется булева формула, состоящая из пяти скобок. В данном случае переменные ветвления выбираются по порядковым номерам переменных. На первом шаге выбирается переменная x_1 и ей присваивается значение 0. При этом становится выполненной вторая скобка, которую можно исключить из формулы для дальнейшего поиска. Также переменная x_1 исчезает из первой, третьей и четвертой скобок, так как она уже не может сделать эти скобки выполненными.

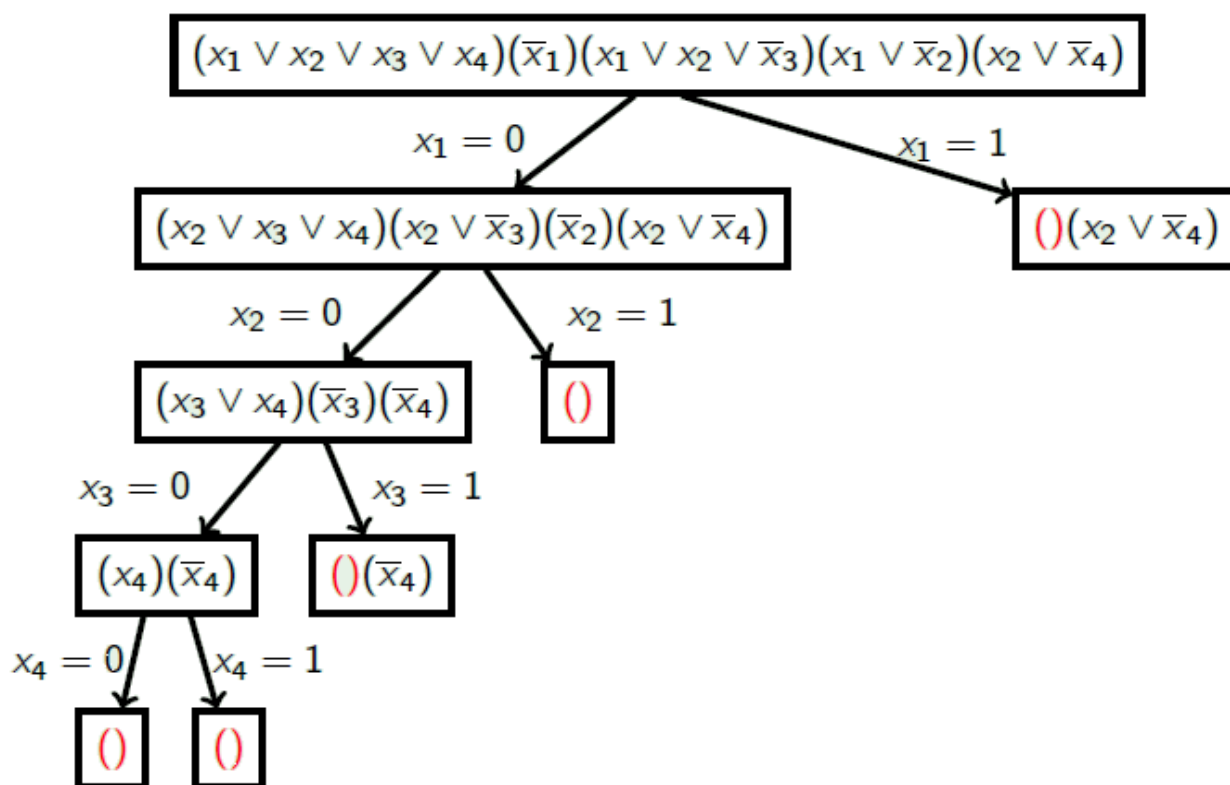


Рисунок 2. Пример работы DPLL алгоритма

Далее, аналогично выбираются переменные $x_2 = 0$ и $x_3 = 0$, и от формулы остаются только две скобки, одна из которых содержит переменную x_4 , а другая – ее отрицание. Полученная формула заведомо является невыполнимой и алгоритм вынужден делать шаг назад и проверять другую ветвь дерева решений, присваивая переменной x_3 значение 1. После этого при работе алгоритма снова возникают несколько конфликтов и алгоритму приходится возвращаться до переменной x_1 , меняя ее значение на 1. Затем полученная формула сразу содержит пустую скобку, и так как алгоритм проверил все пространство решений, то он делает вывод о том, что исходная формула является невыполнимой.

2.3 Машинное обучение и SAT

На данный момент существуют работы, в которых машинное обучение различным образом используется при решении SAT задач. Так, например, одной из самых интересных работ является проект SATzilla [5]. SATzilla – это

портфолио алгоритмов для решения SAT, в котором техники машинного обучения используются для выбора самого быстрого алгоритма для конкретной булевой формулы. В этой же работе систематизированы группы признаков SAT формул, используемых при обучении моделей. Машинное обучение применительно к решению булевых формул с кванторами исследовали авторы статьи [7]. В работе [12] машинное обучение используется для выбора критерия ветвления в DPLL алгоритме. Авторы статьи [8] рассматривают SAT как задачу бинарной классификации и добиваются точности более 70%. Графовое представление SAT формул с последующим применением машинного обучения используется для выявления схожих структур в формулах в работе [9], а также для применения нейронных сетей в решении SAT [10].

2.3.1 Значимые признаки SAT формул

В проекте SATzilla для обучения моделей используется целый набор признаков, которые авторы разделяют на семь групп. Практически все работы, посвященные машинному обучению в решении SAT задач, опираются на эти признаки или пытаются построить новые на их основе, как например в работе [11]. В данной работе для построения пространства признаков используются первые пять групп признаков из проекта SATzilla. Таблица 1 систематизирует эти признаки.

Первая категория ясна из названия и просто отражает количество переменных, количество скобок в булевой формуле и их отношение. Две следующих категории относятся к двум различным графовым представлениям SAT формул. Граф Переменных-Скобок – это граф, в котором вершинами являются переменные и скобки формулы, а ребра находятся между переменными и скобками, в которые эти переменные входят. Для такого графа вычисляются степени вершин-Переменных и вершин-Скобок и берутся их различные статистики распределения. Второй граф – граф-Переменных – граф, вершинами которого являются переменные, а ребра проводятся между

теми переменными, которые хотя бы один раз встретились вместе в одной скобке. Для этого графа, аналогично, вычисляются статистики распределения степеней вершин.

Таблица 1. Значимые признаки булевых формул

Признаки размера 1. Количество скобок (c) 2. Количество переменных (v) 3. Отношение c/v	Признаки баланса 18-20. Отношение количества положительных и отрицательных литералов в каждой скобке: среднее, коэффициент вариации и энтропия. 21-25. Отношение количества положительных и отрицательных вхождений в формулу для каждой переменной: среднее, коэффициент вариации, минимум, максимум и энтропия. 26-27. Доля скобок с двумя и с тремя переменными
Признаки графа Переменных-Скобок 4-8. Статистика распределения степеней вершин-Переменных: среднее, коэффициент вариации, минимум, максимум и энтропия. 9-13. Статистика распределения степеней вершин-Скобок: среднее, коэффициент вариации, минимум, максимум и энтропия.	
Признаки графа Переменных 4-17. Статистика распределения степеней вершин: среднее, коэффициент вариации, минимум и максимум.	Близость к формуле Хорна 28. Доля скобок Хорна 29-33. Количество вхождений в скобки Хорна для каждой переменной: среднее, коэффициент вариации, минимум, максимум и энтропия.

Признаки баланса характеризуют распределение положительных и отрицательных переменных в SAT формуле. Последняя группа признаков описывает близость булевой формулы к формуле Хорна. Эта группа признаков вычисляет различные статистики, относящиеся к скобкам из формулы, которые представляют собой дизъюнкт Хорна. Дизъюнкт Хорна – это дизъюнкция литералов, в которой присутствует не более одного литерала без отрицания. В статье [8] утверждается, что данная группа признаков является важной при бинарной классификации, так как близость к формуле Хорна отражает то, насколько возможно решить SAT за полиномиальное время с использованием стандартного вывода.

Глава 3

Идея и план разработки

3.1 Идея применения классификатора

Данная работа основывается на результатах исследований по различному применению машинного обучения в решении SAT задач. Однако, в отличие от существующих работ, здесь предлагается использовать техники машинного обучения для увеличения производительность конкретного SAT алгоритма – DPLL.

Алгоритм DPLL использует процедуру ветвления, в которой выбирается переменная ветвления. Выбранной логической переменной присваивается значение 0 или 1, и затем алгоритм продолжает свое выполнение рекурсивно. Когда возникает конфликт, то есть появляется пустая скобка, алгоритм вынужден делать шаг назад и проверять вторую ветвь дерева решений. Хотя алгоритм DPLL и использует различные эвристические правила для выбора переменной ветвления, он не принимает никакого «умного» решения о том, какое значение сперва присвоить этой переменной. Поэтому предлагается использовать бинарный классификатор для выбора значения переменной ветвления, чтобы уменьшить количество конфликтов, возникающих при работе алгоритма, и соответственно также время его работы.

Эвристика предложенного подхода основывается на предположении о том, что, если в каждой точке ветвления алгоритма присваивать переменной ветвления значение, которое с большей вероятностью ведет к решению, тогда решение должно быть найдено относительно быстро. Таким образом, увеличивается производительность DPLL алгоритма для поиска решений выполнимых формул. Далее описывается алгоритм выбора значения переменной ветвления, который использует вероятности, возвращаемые бинарным классификатором.

3.2 Алгоритм выбора значения переменной ветвления

Для того, чтобы определить значение переменной ветвления, которое вероятней всего ведет к решению, предлагается изначально в формуле зафиксировать за переменной ветвления сначала значение 1 и использовать бинарный классификатор для определения вероятности того, что упрощенная формула, полученная после фиксации значения одной переменной, является выполнимой. Аналогично, необходимо зафиксировать за переменной ветвления значение 0 и снова использовать классификатор для определения вероятности, что упрощенная формула является выполнимой. Затем, можно принять решения о присвоении значения, для которого вероятность получилась больше.

Однако, такая упрощенная стратегия принятия решения о присвоении значения переменной может быть модифицирована. Алгоритм 2 описывает общий подход к выбору значения переменной ветвления, предложенный в данной работе. Предлагается после фиксации значения переменной ветвления зафиксировать значения еще одной, следующей переменной, случайным образом и вычислить вероятность выполнимости для такой формулы после фиксации второй переменной. После этого вероятность того, что выбранное значение переменной ветвления ведет к выполнимости формулы можно считать, как произведение вероятности после фиксации значения переменной ветвления и вероятности после фиксации второй переменной. Так как выбор второй фиксированной переменной случаен, то можно сделать несколько таких попыток (*trials*) случайной фиксации значения второй переменной и соответственно получить несколько значений вероятностей. Решение о присвоении 0 или 1 переменной ветвления предлагается определять по наибольшему среднему значению по нескольким таким попыткам. Более того, предлагается увеличить «глубину» (*depth*) фиксации до двух случайных переменных и вычислять вероятность выполнимости формулы после присвоения значения второй случайной переменной.

Алгоритм 2. *decide_var(F , clf , x , $trials$, $depth$)*

```
//массив для хранения вероятностей при фиксации переменной ветвления значения 1
pos_probas = array[]
//массив для хранения вероятностей при фиксации переменной ветвления значения 0
neg_probas = array[]

pos_F =  $F|_x$  //фиксация значения 1 для переменной ветвления
neg_F =  $F|_{\neg x}$  //фиксация значения 0 для переменной ветвления

//вычисление вероятностей выполнимости формулы при фиксированной 1
for i from 1 to trials //цикл количества случайных проб
    curr_F = pos_F //сброс текущей формулы
    curr_proba = 1 //сброс текущей вероятности выполнимости
    for j from 0 to depth //цикл глубины фиксации
        //предсказание вероятности выполнимости текущей формулы
        curr_proba = curr_proba *  $clf.predict(curr_F)$ 
        //фиксация случайной переменной
        curr_F = curr_F|random
    pos_probas.put(curr_proba) //добавление результата пробы в массив

//вычисление вероятностей выполнимости формулы при фиксированном 0
for i from 1 to trials //цикл количества случайных проб
    curr_F = neg_F //сброс текущей формулы
    curr_proba = 1 //сброс текущей вероятности выполнимости
    for j from 0 to depth //цикл глубины фиксации
        //предсказание вероятности выполнимости текущей формулы
        curr_proba = curr_proba *  $clf.predict(curr_F)$ 
        //фиксация случайной переменной
        curr_F = curr_F|random
    neg_probas.put(curr_proba) //добавление результата пробы в массив

//сравнение средних значений
if mean(pos_probas) > mean(neg_probas)
    return 1
else
    return 0
```

Алгоритм 2 в работе реализует функция *decide_var*, которая принимает на вход формулу F в формате CNF, классификатор clf , используемый при принятии решения, саму переменную x , решение о присвоении значения которой необходимо сделать, а также два дополнительных параметра: *trials* – количество проб фиксации случайных переменных для измерения

вероятностей и *depth* – глубину фиксации случайных переменных (в работе используются значения 0, 1 и 2).

3.3 Общая схема внедрения машинного обучения

Общая схема внедрения машинного обучения в алгоритм DPLL показана на Рисунке 3. Можно выделить несколько основных этапов машинного обучения в процессе разработки: построение обучающей выборки, вычисление признаков и обучение классификаторов.

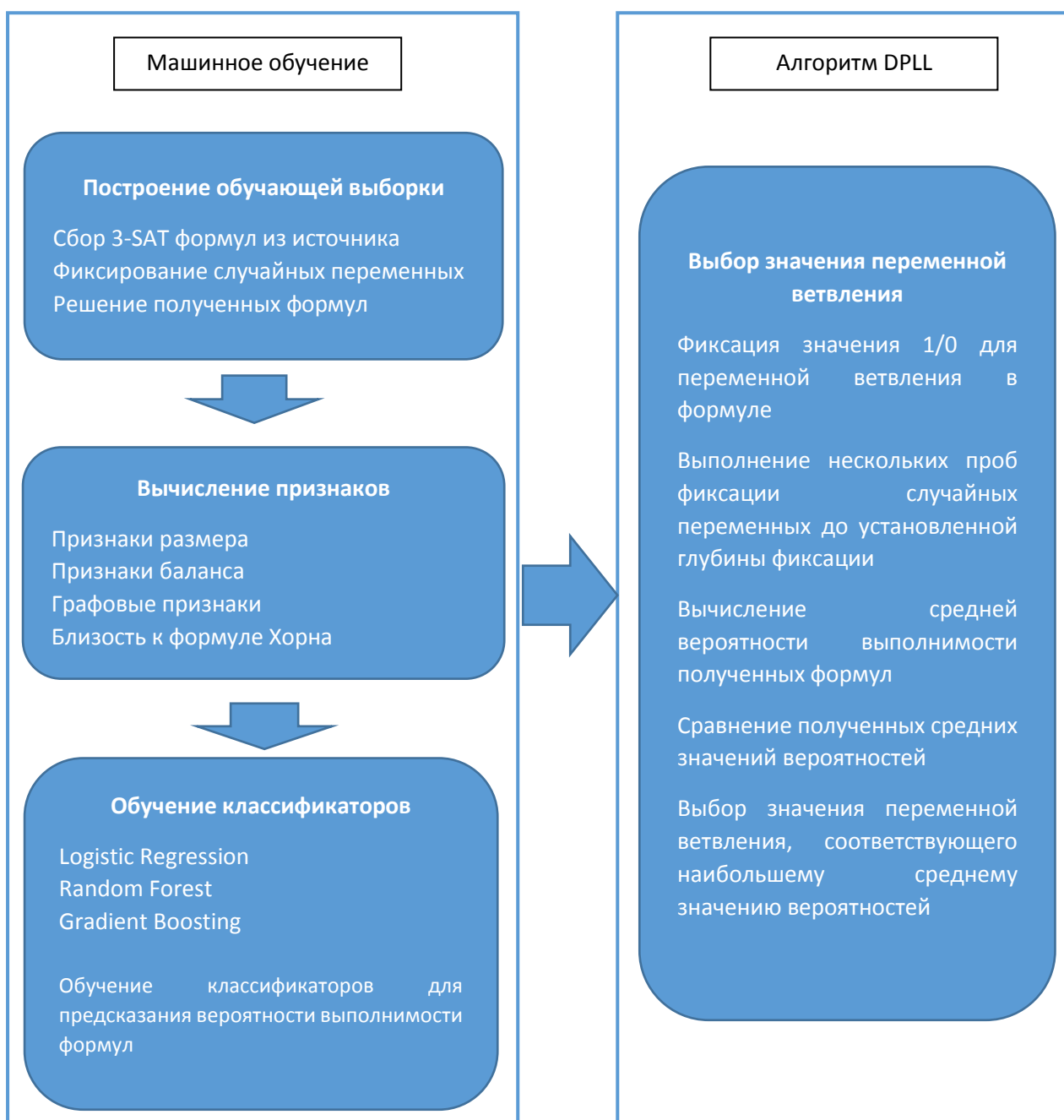


Рисунок 3. Схема внедрения машинного обучения

3.3.1 Построение обучающей выборки

Для обучения и тестирования в данной работе используются 3-SAT формулы из открытой базы SAT формул для проведения соревнований SATLIB [14]. Подробно этот ресурс освещен в статье [2]. Все формулы берутся из раздела, содержащего только формулы из области фазового перехода, описанного в главе 2. Всего используется 9 наборов выполнимых формул с разным количеством переменных: 50, 75, 100, 125, 150, 175, 200, 225 и 250. Обучение и тестирование проводится для каждого набора отдельно.

При построении обучающей выборки для каждой формулы из набора фиксируются значения одной, двух и трех переменных, вычисляются признаки полученных формул и определяется их выполнимость. Для определения выполнимости формул при построении обучающей выборки для определения выполнимости формул используется готовый SAT алгоритм – PicoSAT [13].

3.3.2 Обучение классификаторов

Следующим этапом после построения обучающей выборки идет обучения классификаторов. Сначала предлагается обучение различных стандартных классификаторов (Logistic Regression, Random Forest, Gradient Boosting) и выбор наилучшего для каждого набора формул с использованием кросс-валидации. Затем для выбранного классификатора проводится настройка параметров тоже с использованием кросс-валидации. Проверка производительности проводится с использованием тестовой выборки. Описание алгоритмов машинного обучения не входит в данную работу, так это не является ее целью, тем более что их подробный обзор может быть найден в любой стандартной литературе по машинному обучению, например, в [6].

Глава 4

Реализация

4.1 Техническая спецификация

Для реализации всего проекта используется язык программирования Python 2. Вся разработка ведется в операционной системе Ubuntu 16.04 LTS. Для написания кода алгоритмов и вспомогательных функций используется интегрированная среда разработки PyCharm Community Edition 2016.3.3. Для проведения экспериментов используется пакет Jupyter Notebook. Стандартные классификаторы берутся из пакета scikit-learn.

Характеристики машины:

- CPU: Intel Core i5-5300U 2.30GHz x 4
- RAM: 8GB

4.2 Описание классов

В проекте используются два класса, которые инкапсулируют в себя весь необходимый функционал. Класс CNF служит для чтения формулы из файла, вычисления признаков и прочего. Класс DPLL содержит в себе функцию, реализующие различные модификации алгоритма, а также функции, возвращающие различную статистику выполнения алгоритма, например, время работы алгоритма, количество конфликтов и другое.

4.2.1 Класс CNF

Класс CNF служит для работы с булевыми формулами в стандартном формате CNF. Все SAT формулы в данной работе являются объектами этого класса. Инициализация SAT формулы – объекта класса CNF может происходить двумя способами. Булева формула может читаться из файла формата DIMACS. DIMACS – это стандартизованный формат файлов для

записи SAT формул в форме CNF. Пример такого файла изображен на Рисунке 4.

```
c This Formular is generated by mcnf
c
c   horn? no
c   forced? no
c   mixed sat? no
c   clause length = 3
c
p cnf 4 5
1 2 3 4 0
-1 0
1 2 -3 0
1 -2 0
2 -4 0
%
0
```

Рисунок 4. DIMACS формат

В этом формате строки соответствуют переменным, находящимся в одной скобке в SAT формуле. Знак '-' означает отрицание переменной. Вторым вариантом инициализации SAT формулы – чтение из списка скобок.

Объекты класса CNF имеют методы, соответствующие вычислению признаков SAT формулы по группам. Метод *get_features* реализует вычисление всех признаков для данной SAT формулы. Метод *set_var* фиксирует значение заданной переменной в SAT формуле. Если переменная не указана, то фиксируется случайное значение случайно выбранной переменной.

4.2.2 Класс DPLL

Объектом класса DPLL является непосредственно сам SAT решатель. Инициализация объекта происходит путем передачи списка параметров, состоящего из SAT формулы для решения в виде объекта класса CNF, критерия ветвления и флага использования классификатора. Если классификатор используется, то дополнительно нужно задать параметры *trials* и *depth* для алгоритма выбора значения переменной ветвления с использованием классификатора.

Сам алгоритм DPLL реализован в методе *dpll* данного класса. Подфункция *UnitPropagate* из Алгоритма 1 реализована в методе *unit_propagate*. Метод *solve* класса DPLL запускает в работу алгоритм, подсчитывает количество конфликтов, возникших при выполнении алгоритма, а также вычисляет общее время выполнения алгоритма DPLL для заданной SAT формулы.

4.3 Описание вспомогательных функций

В данном проекте используются три вспомогательных функции. Функция *decide_var* реализует Алгоритм 2 выбора значения переменной ветвления. Функция *get_backbones* определяет базовые переменные для данной SAT формулы и их значения. Базовыми переменными являются те переменные, которые имеют одно и то же значение во всех решениях для конкретной SAT формулы. Функция *preprocessing* предсказывает значения для всех переменных посредством использования функции *decide_var*, а также вычисляет точность задания базовых переменных и собственное время выполнения.

Глава 5

Экспериментальные результаты и анализ

В работе используются 9 различных наборов формул с разным количеством переменных. Название каждого набора содержит информацию о количестве переменных и скобок в формулах данного набора (n_m , где n – это количество переменных, m – это количество скобок). Каждый набор состоит из 100 SAT формул. Для проведения экспериментов каждый набор SAT формул был разбит на обучающий и тестовый наборы в соотношении 70:30. Эксперименты проводились над каждым набором SAT формул отдельно в целях последующего анализа и сравнения результатов для SAT формул с разным количеством переменных.

5.1 Машинное обучение

Результаты экспериментальной части в данном разделе представлены на примере набора SAT формул с 75 переменными.

Предварительным этапом перед обучением классификаторов необходимо было построить обучающую выборку описанным в главе 3 способом. Таблица 2 содержит полученное в результате построения количество положительных и отрицательных примеров для каждого из набора формул.

Таблица 2. Размер обучающей выборки

наборы:	50_218	75_325	100_430	125_538	150_645	175_753	200_860	225_960	250_1065
SAT	5400	5571	5630	5730	6674	6360	6047	6215	6384
UNSAT	5845	5675	5619	5519	4575	4890	5201	5032	4866
ALL	11245	11246	11249	11249	11249	11250	11248	11247	11250

В данной работе в качестве исследуемых классификаторов были предложены три различных модели. Random Forest и Gradient Boosting были

выбраны по причине того, что они показали наилучшую точность при бинарной классификации в работе [8]. Также дополнительно была исследована производительность модели Logistic Regression.

Оценка качества классификаторов выполнялась с использованием кросс-валидации с разбиением обучающей выборки на 10 частей. Так как для некоторых наборов формул выборка получилась немного несбалансированной то наряду с точностью классификаторов в работе оценивались ROC-кривые и площади под ними для каждого классификатора. Помимо этого, при кросс-валидации использовалась функция `StratifiedKFold` из пакета `scikit-learn`, которая разбивает данные на части, сохраняя изначальное соотношение положительных и отрицательных примеров в выборке. Пример ROC-кривой для набора формул с количеством переменных $N = 75$ представлен на Рисунке 5.

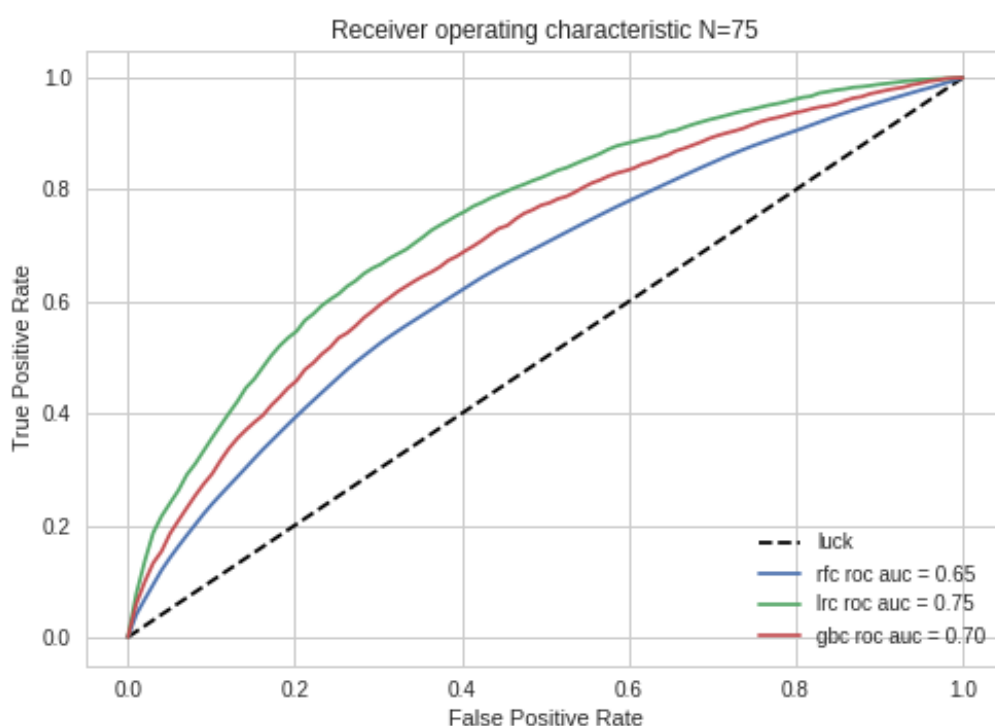


Рисунок 5. Пример ROC-кривой обучения классификаторов

Точность классификаторов на обучающей выборке с использованием кросс-валидации представлена в Таблице 3. В работе используются следующие сокращения названий классификаторов: Random Forest – rfc, Logistic Regression – lrc, Gradient Boosting – gbc.

Таблица 3. Точность классификаторов на кросс-валидации

наборы:	50_218	75_325	100_430	125_538	150_645	175_753	200_860	225_960	250_1065
rfc	0.65	0.61	0.61	0.59	0.62	0.58	0.59	0.60	0.59
lrc	0.70	0.68	0.67	0.65	0.68	0.66	0.66	0.66	0.68
gbc	0.68	0.64	0.65	0.60	0.64	0.60	0.60	0.64	0.62

Модель Logistic Regression показала лучший результат на всех наборах данных как по точности, так и по площади под ROC-кривой. Поэтому эта модель используется далее в качестве основного классификатора при тестировании работы SAT алгоритма. Предварительно были настроены коэффициенты регуляризации данной модели для каждого набора формул отдельно также с использованием кросс-валидации с разбиением обучающей выборки на 10 частей. На Рисунке 6 на примере набора формул с N=75 показана зависимость точности модели Logistic Regression от варьирования параметра коэффициента регуляризации. Здесь можно заметить только то, что с увеличением количества переменных в формулах приходилось увеличивать коэффициент регуляризации.

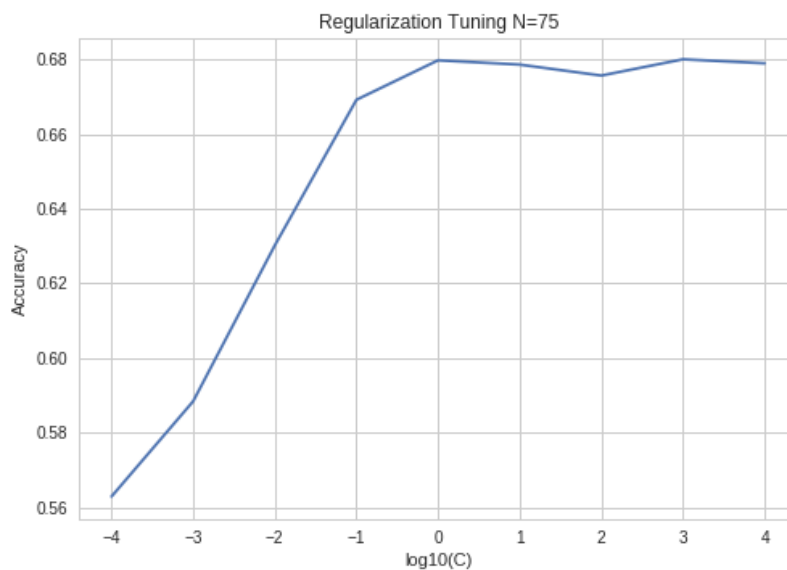


Рисунок 6. Пример зависимости точности модели Logistic Regression от коэффициента регуляризации

На Рисунке 7 показан пример распределения вероятностей, выдаваемых классификатором Logistic Regression для тестового набора формул с $N=75$. На этом рисунке видна зависимость точности классификатора от уверенности предсказания. Это можно проследить путем сравнения высоты столбцов правильных и ошибочных предсказаний для каждой предсказанной вероятности SAT. Когда предсказанная вероятность SAT формулы близка к 0 или 1, классификатор практически не ошибается. Но для вероятностей близких к 0,5 точность классификатора падает, приближаясь к значению 0,5, что соответствует случайному угадыванию.

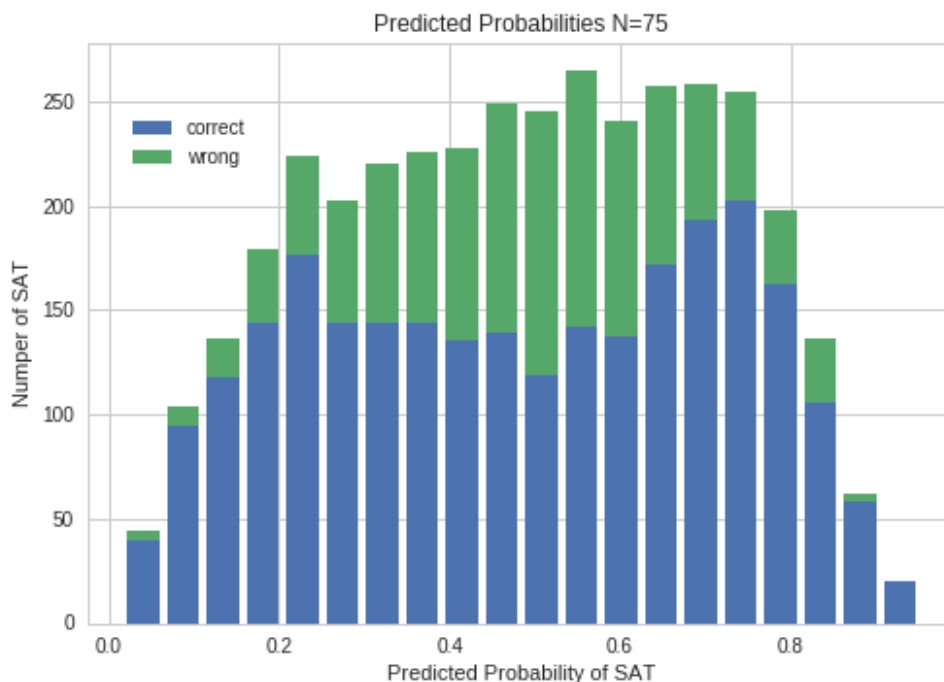


Рисунок 7. Пример распределения вероятностей классификатора

5.2 Тестирование DPLL алгоритма

С точки зрения производительности DPLL алгоритма с использованием классификатора интересными показателями являются доля правильно предсказанных значений так называемых базовых переменных, количество конфликтов, возникших в работе алгоритма, а также непосредственно само время выполнения алгоритма в сравнении со стандартной версией. При проведении экспериментов в DPLL алгоритме в качестве критерия ветвления использовался самый эффективный – MOMS. Критерии RANDOM и MAXO используются дополнительно в заключительном эксперименте в целях сравнения.

Первым этапом в данной работе было необходимо подобрать оптимальные параметры алгоритма выбора значения переменной ветвления, а именно глубину фиксирования переменных – depth и количество случайных проб – trials. Параметр depth варьировался среди значений 0, 1, и 2, параметр trials – 1, 3, 5, 7. Подбор оптимальных значений опирался на соображения о

минимальном среднем числе конфликтов в работе DPLL алгоритма для данных параметров. На Рисунках 8 и 9 представлены результаты эксперимента по запуску алгоритма DPLL с различными параметрами для набора формул с $N=50$ и $N=75$ соответственно.

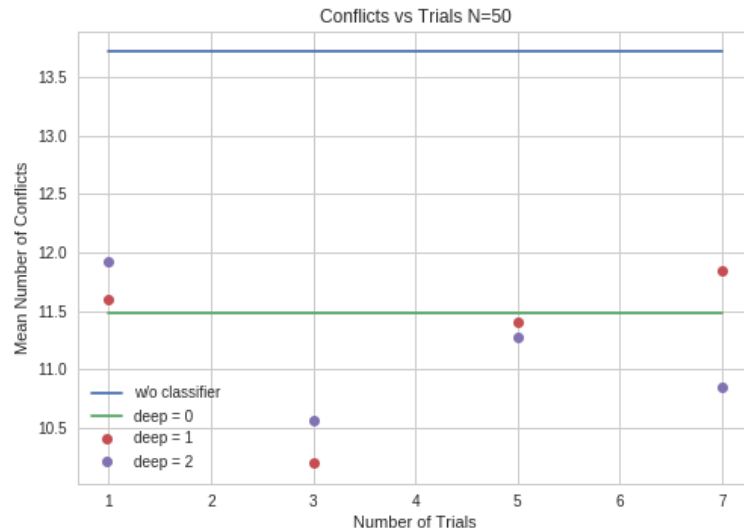


Рисунок 8. Среднее количество конфликтов для различных значений параметров алгоритма *decide_var* при $N=50$

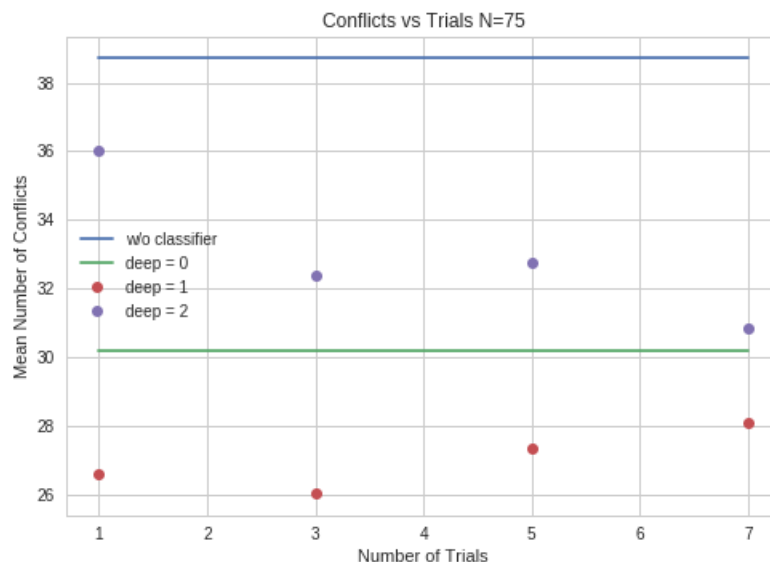


Рисунок 9. Среднее количество конфликтов для различных значений параметров алгоритма *decide_var* при $N=75$

На этих рисунках синяя линия соответствует уровню среднего числа конфликтов алгоритма DPLL без использования классификаторов. Естественнo, что это значение не зависит от параметра *trials*. Зеленая линия соответствует уровню среднего числа конфликтов алгоритма DPLL с использованием классификатора с параметром *depth* = 0. Аналогично, это значение не зависит от количества случайных проб. А вот красные и фиолетовые точки на графиках меняются с изменением значения параметра *trials* и соответствуют алгоритму DPLL с параметрами *depth* = 1 и *depth* = 2. На разных наборах формул характер этих зависимостей разный, но практически везде, за редким исключением, наименьшее среднее число конфликтов наблюдается при параметрах *depth* = 1 и *trials* = 3. Поэтому далее в работе используются эти значения параметров.

Дополнительно с использованием функции *preprocessing* была измерена доля правильно предсказанных значения базовых переменных для каждого набора формул. Результаты, представленные на Рисунке 10, говорят о высокой предсказательной точности базовых переменных. Точность предсказания значений базовых переменных не падает ниже 70% с увеличением количества переменных в формуле.

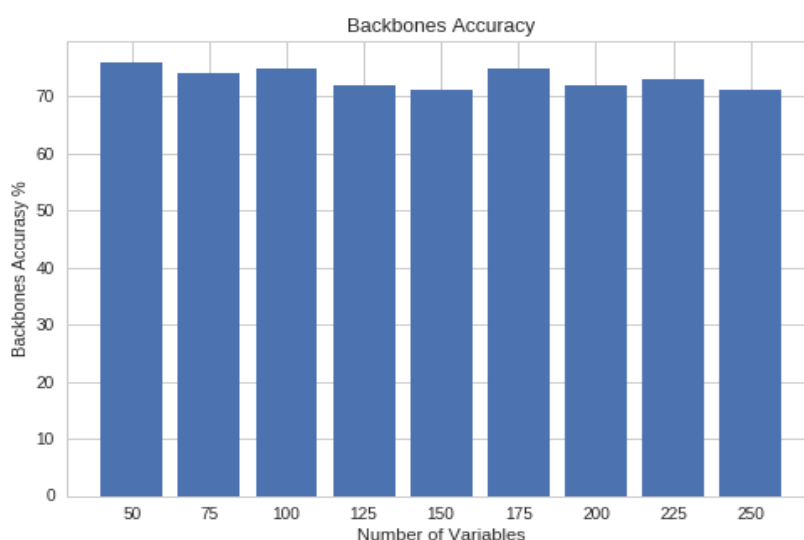


Рисунок 10. Точность предсказания значений базовых переменных

Целью данной работы была оценка производительности SAT алгоритма с использованием бинарного классификатора в сравнении с его базовой версией. Поэтому в конце был проведен эксперимент с измерением времени выполнения алгоритма с использованием классификатора и без него. На Рисунке 11 представлены два графика (синий – без классификатора, зеленый – с классификатором) зависимости среднего времени выполнения алгоритма DPLL с критерием ветвления MOMS от числа переменных в формуле. Использование бинарного классификатора дает прирост производительности для формул с количеством переменных больше, чем 175.

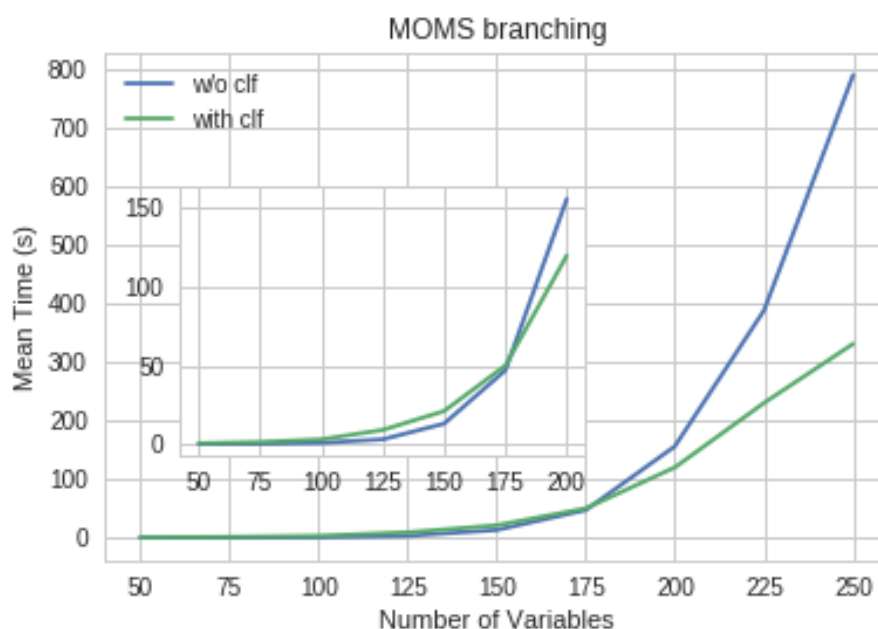


Рисунок 11. Производительность алгоритма DPLL MOMS

Аналогичные эксперименты были проведены для алгоритма DPLL с критериями ветвления MAXO и RANDOM. Результаты этих экспериментов представлены на Рисунках 12 и 13 соответственно.

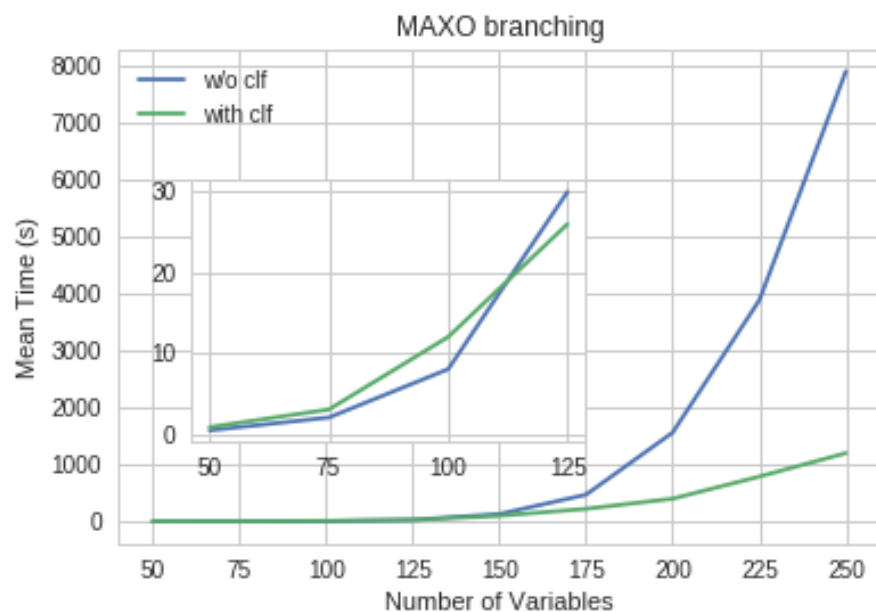


Рисунок 12. Производительность алгоритма DPLL MAXO

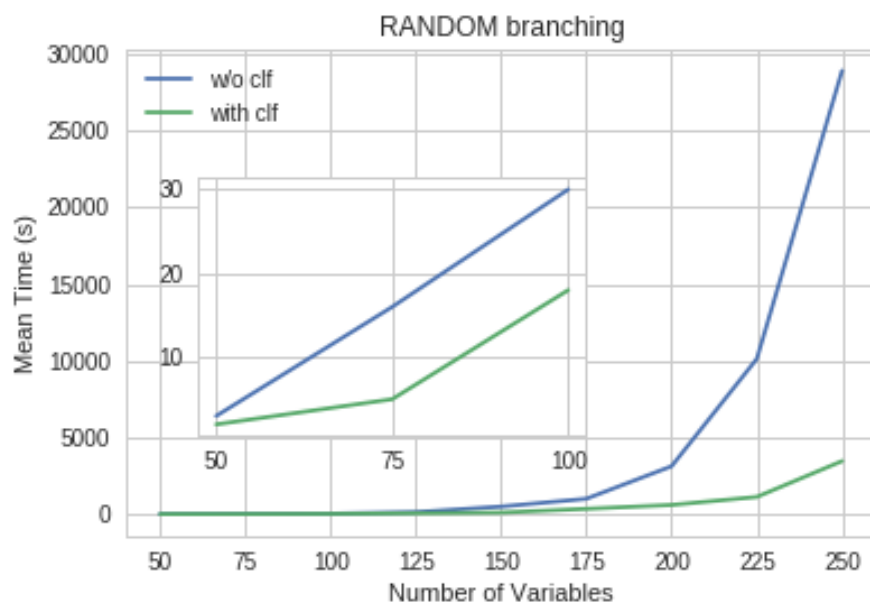


Рисунок 13. Производительность алгоритма DPLL RANDOM

Для алгоритма DPLL с критерием ветвления MAXO порог, при котором классификатор начинает давать увеличение производительности, меньше, чем у алгоритма DPLL с критерием ветвления MOMS, и соответствует количеству переменных $N = 100$. Для алгоритма DPLL с критерием ветвления RANDOM классификатор уменьшает время работы для всех наборов формул. Это объясняется тем, что при критерии ветвления RANDOM правильный выбор значения переменной ветвления позволяет компенсировать неразумный выбор этой переменной, направляя алгоритм по нужной ветви дерева решений. А при критерии ветвления MOMS алгоритм и так делает разумный выбор переменной ветвления, поэтому вклад работы классификатора по предсказанию ее значения нивелируется временными затратами на саму процедуру предсказания. Работа классификатора становится заметной только при увеличении числа переменных в формулах, так как в этом случае временные затраты на предсказание значения переменной ветвления становятся меньше, чем время, которое тратит алгоритм на разрешение большего числа конфликтов без применения классификатора.

Глава 6

Заключение

6.1 Обзор проделанной работы

В данной работе на примере SAT проблемы было продемонстрировано использование классического алгоритма систематического поиска в сочетании с методами машинного обучения. В начале была подробно освещена проблема выполнимости булевых формул в целом. Затем был поэтапно изложен один из базовых SAT алгоритмов – DPLL. Наряду с этим, был проведен детальный обзор существующих методик по применению машинного обучения к решению задачи SAT. В основной части работы был разработан и предложен метод использования бинарного классификатора в алгоритме DPLL. Этот метод базируется на уже имеющихся результатах работ по применению машинного обучения в SAT задачах, а также на предложенном в работе алгоритме выбора значения переменной ветвления с использованием бинарного классификатора. В работе предложен алгоритм выбора значения переменной ветвления с использованием бинарного классификатора. Экспериментальная часть показала, что достигнутое качество обучения классификаторов, а также предложенный в работе подход улучшают работу DPLL алгоритма. Точность классификаторов на кросс-валидации достигала значения 0,7. Также экспериментальная часть содержит раздел, предназначенный для выбора оптимальных значений параметров предложенного алгоритма. Осмысленность методики по применению машинного обучения демонстрируют результаты эксперимента по предсказанию значений базовых переменных. Точность предсказания их значений на всех наборах формул больше 70%. Были исследованы и обозначены условия повышения производительности. Так было выявлено, что применение бинарного классификатора по описанной в работе методике в

алгоритме DPLL MOMS имеет смысл для формул с $N > 175$, в алгоритме DPLL MAXO – с $N > 100$, в алгоритме DPLL RANDOM – с $N > 50$.

6.2 Программный код проекта

Отдельным результатом данного проекта является хорошо продуманная и структурированная реализация кода, которая позволяет в дальнейшем строить исследование уже на основе данной работы. По окончании исследования реализованные в работе программные файлы были выложены в открытый доступ в личный GitHub репозиторий по адресу: https://github.com/mosin26/master_thesis.

6.3 Направления дальнейших исследований

К сожалению, предложенный в работе подход по использованию бинарного классификатора в DPLL алгоритме при выборе значения переменной ветвления дает прирост производительности по времени работы алгоритма только на пространстве выполнимых SAT формул. Критерий выбора самой переменной ветвления (например, MOMS, MAXO или RANDOM), конечно же, влияет на скорость работы DPLL алгоритма как в случае выполнимых формул, так и в случае невыполнимых формул, потому что правильный выбор переменной ветвления позволяет быстрее прийти к решению или доказательству его отсутствия. Однако, для невыполнимых SAT формул алгоритм DPLL, выбрав конкретную переменную ветвления, обязан проверить обе ветви дерева решений, соответствующие двум различным значениям. Поэтому выбор того, какую из ветвей проверить в первую очередь, не имеет значения в плане времени работы алгоритма в случае невыполнимых булевых формул.

Список дальнейших исследований по данной тематике может включать различного рода задачи по приспособлению предложенной в работе методики применения бинарного классификатора для разных случаев. Например, одной из потенциальных задач может являться аналогичное исследование

возможности применения машинного обучения в неполных SAT алгоритмах, тем более, что в данной работе уже предложена функция *preprocessing*, которая может использоваться для генерации начального набора значений переменных. Другим направлением продолжения данной работы может быть модификация предложенного подхода для увеличения производительности работы DPLL алгоритма в случае невыполнимых булевых формул.

Список литературы

- [1] Carla P. Gomes, Henry Kautz, Ashish Sabharwal, and Bart Selman. Satisfiability Solvers. In *Handbook of Knowledge Representation*, pages 89–134, 2008.
- [2] Holger H. Hoos and Thomas Stutzle. *Satlib: An online resource for research on sat*. pages 283–292.
- [3] M. N. Velev and R. E. Bryant. Effective use of Boolean satisfiability procedures in the formal verification of superscalar and vliw microprocessors. *J. Symb. Comput.*, 35(2):73–106, 2003.
- [4] Stephen A. Cook. The complexity of theorem-proving procedures. In *STOC '71: Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158, New York, NY, USA, 1971. ACM.
- [5] Lin Xu, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Satzilla-07: The design and analysis of an algorithm portfolio for sat. In *Principles and Practice of Constraint Programming (CP-07)*, Lecture Notes in Computer Science 4741, pages 712–727. Springer Berlin, 2007.
- [6] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.
- [7] Horst Samulowitz and Roland Memisevic. Learning to Solve QBF. In *AAAI'07 Proceedings of the 22nd national conference on Artificial intelligence - Volume 1*, pages 255–260, Vancouver, British Columbia, 2007.
- [8] David Devlin and Barry O'Sullivan. *Satisfiability as a Classification Problem*. University of College Cork, Ireland.
- [9] Carlos Ansotegui, Maria Luisa Bonet, Jesus Giráldez-Cru, and Jordi Levy. *Community Structure in Industrial SAT Instances*. 2016.
- [10] Benedikt Bünz, Matthew Lamm. *Graph Neural Networks and Boolean Satisfiability*. 2017.

- [11] Enrique Alfonso, Norbert Manthey. New CNF Features and Formula Classification. In *Daniel Le Berre, eds., POS-14, volume 27 of EPiC Series*, 57–71, 2014.
- [12] Michail G. Lagoudakis, Michael L. Littman. *Learning to Select Branching Rules in the DPLL Procedure for Satisfiability*. 2001.
- [13] Armin Biere. PicoSAT Essentials. *Journal on Satisfiability, Boolean Modeling and Computation (JSAT)*, vol. 4, pages 75-97, Delft University, 2008.
- [14] SATLIB-Benchmark <http://www.cs.ubc.ca/~hoos/SATLIB/benchm.html> 2017.
- [15] SAT Competitions <http://www.satcompetition.org/> 2017.
- [16] Lin Xu, Holger H. Hoos, Kevin Leyton-Brown. Predicting satisfiability at the phase transition. In *AAAI'12 Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, pages 584–590, Toronto, Ontario, Canada, 2012. ACM.
- [17] Ming Ouyang. *How good are branching rules in DPLL?* 1996.
- [18] Stefan Harmeling. *Solving Satisfiability Problems with Genetic Algorithms*. 2000.
- [19] Peter Cheeseman, Bob Kanefsky, and William M. Taylor. Where the really hard problems are. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence, IJCAI-91, Sidney, Australia*, pages 331–337, 1991.