Минобрнауки России
Федеральное государственное бюджетное учреждение науки
Институт динамики систем и теории управления имени В.М. Матросова
Сибирского отделения Российской академии наук
(ИДСТУ СО РАН)

На правах рукописи

Кондратьев Виктор Сергеевич

Методы обработки конфликтных ограничений в алгоритмах решения SAT, основанных на нехронологическом бэктрекинге

09.06.01 — Информатика и вычислительная техника

1.2.2 — Математическое моделирование, численные методы и комплексы программ

НАУЧНЫЙ ДОКЛАД
об основных результатах научно-квалификационной работы (диссертации)
на соискание ученой степени
кандидата технических наук

Иркутск — 2023

Работа выполнена в лаборатории 6.2. Логических и оптимизационных методов анализа сложных систем отделения 6. Методов невыпуклой и комбинаторной оптимизации ИДСТУ СО РАН.

Научный руководитель: **Семёнов Александр Анатольевич,**
к.т.н., доцент,
в.н.с., ИДСТУ СО РАН

Рецензенты: **Ульянцев Владимир Игоревич,**
к.т.н.,
с.н.с., Университет ИТМО

Фёдоров Роман Константинович,
к.т.н.,
в.н.с., зав. лаб. 4.1, ИДСТУ СО РАН

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Актуальность темы. В работе изучаются и разрабатываются алгоритмы решения проблемы булевой выполнимости (Boolean Satisfiability Problem, SAT), которую можно рассматривать как простейший случай известной проблемы удовлетворения ограничений (Constraint Satisfaction Problem, CSP).

Проблема удовлетворения ограничений — это вычислительная задача, в которой целью является поиск решения, удовлетворяющего множеству ограничений. Впервые данная проблема была сформулирована У. Монтанари в его работе, связанной с распознаванием образов[1], после чего ее исследование было продолжено многими учеными, среди которых можно выделить Р. Дехтер, С. Рассела, П. Норвига, М. Гэри, Д. Джонсона, Т. Уолша и многих других.

CSP состоит из множества переменных, доменов для каждой переменной и набора ограничений, которые определяют допустимые комбинации значений для переменных. Задача состоит в том, чтобы для заданной системы ограничений ответить на вопрос о ее совместности, то есть о существовании решения. В том случае, когда ответ на этот вопрос “да”, обычно требуется найти произвольное решение. Задача CSP с конечными доменами (множествами значений переменных) является NP-полной в распознавательной постановке и NP-трудной, если требуется найти ее решение. Несмотря на высокую теоретическую сложность, интерес к разработке практических вычислительных алгоритмов для CSP и различных ее частных случаев неуклонно растет. Хотя по определению NP-полной задачи к любой такой задаче сводится любая задача из класса NP, преимущество задачи CSP состоит в том, что большинство комбинаторных задач может быть сведено к ней намного проще и естественнее, чем к другим NP-полным задачам.

Частным случаем CSP является задача булевой выполнимости (SAT), в которой домены переменных представлены множеством {0,1}, а ограничения представлены в виде наборов литералов, называемых дизъюнктами. Формулируется SAT задача следующим образом: по заданной булевой формуле определить, является ли данная формула выполнимой. Булева формула при решении SAT обычно представлена в конъюнктивной нормальной форме (КНФ). SAT является одной из наиболее изучаемых и важных задач в теории вычислительной сложности. Она широко применяется в различных областях, таких как теория алгоритмов, формальные методы верификации программного обеспечения, проектирование и верификация аппаратуры, искусственный интеллект и многих других.

[1]Montanari, U. Networks of constraints: Fundamental properties and applications to picture processing / U. Montanari // Information Sciences. — 1974. — Vol. 7. — P. 95–132.

Как задача поиска SAT является NP-трудной задачей, что означает, что не существует эффективных алгоритмов, которые могут решить эту задачу для любой входной формулы при условии, что $P \neq NP$. Однако существуют эффективные алгоритмы для некоторых подклассов SAT, а также постоянно расширяется множество эвристик, дополняющих базовые алгоритмы решения SAT, делая эти алгоритмы применимыми ко все более обширным классам практических задач.

Тот факт, что SAT допускает естественные эвристические алгоритмы, делает ее привлекательной задачей для сведения к ней комбинаторных задач из перечисленных выше областей. Согласно теореме Кука–Левина[2, 3] любая задача класса NP может быть эффективно (за полиномиальное время) сведена к SAT в ее распознавательном варианте. Это означает, что умение эффективно решать SAT может помочь в решении любой другой задачи из класса NP.

В настоящее время существует множество методов решения SAT, однако лишь некоторые из них получили широкое признание. В практических приложениях SAT доминируют алгоритмы, которые традиционно относятся к вычислительной логике (Computational Logic), среди которых стоит выделить алгоритм Девиса–Патнема–Лоджманна–Лавленда (DPLL)[4] и алгоритм Conflict Driven Clause Learning (CDCL)[5]. Также можно отметить алгоритмы, основанные на стратегии Look-Ahead[6].

Наиболее широкое практическое применение получили SAT-решатели, основанные на алгоритме CDCL. Данный алгоритм можно рассматривать как усиление DPLL возможностью использовать память для хранения информации о тупиковых областях пространства поиска. Данная информация хранится в виде специальных ограничений-дизъюнктов, называемых *выученными дизъюнктами* (learnt clauses). Под тупиковой областью понимается такая часть дерева поиска, для которой доказано отсутствие в ней набора, выполняющего рассматриваемую формулу. Вывод о том, что конкретная ветвь в дереве поиска является тупиковой, делается после ситуации, называемой *конфликтом*.

Использование выученных дизъюнктов, помимо сохранения истории поиска, позволяет явно выделить переменные, присвоение конкретных значений которым привело к конфликту. Часто такое знание дает возможность алго-

[2]Cook, S.A. The Complexity of Theorem-Proving Procedures / S.A. Cook // STOC. — ACM, 1971. — P. 151–158.

[3]Levin, L. Universal Sequential Search Problems / L. Levin // Problems of Information Transmission. — 1973. — Vol. 9, no. 3. — P. 265–266.

[4]Davis, M. A machine program for theorem-proving / M. Davis, G. Logemann, D.W. Loveland // Commun. ACM. — 1962. — Vol. 5, no. 7. — P. 394–397.

[5]Marques-Silva, J.P. GRASP: a New Search Algorithm for Satisfiability / J.P. Marques-Silva, K.A. Sakallah // Proceedings of ICCAD '96. — 1996. — P. 220–227.

[6]Heule, M.J.H. Look-Ahead Based SAT Solvers / M.J.H. Heule, H. van Maaren // Handbook of Satisfiability / Ed. by A. Biere, M. Heule, H. van Maaren, T. Walsh. — 2nd ed. — IOS Press, 2021. — Vol. 336. — P. 183–212.

ритму вернуться не к предшествующему конфликту назначению переменной, а к существенно более раннему. Такая ситуация называется *нехронологическим бэктрекингом*.

В процессе своей работы современные SAT-решатели, базирующиеся на CDCL, могут порождать колоссальные по объему массивы выученных дизъюнктов, называемые также *конфликтными базами*. Хорошо известно, что при сохранении конфликтной базы на диск (когда она перестает помещаться в оперативной памяти компьютера) производительность SAT-решателя существенным образом снижается. Данную проблему можно решить двумя способами: во-первых, периодически удалять некоторые выученные дизъюнкты, уменьшая таким способом размер конфликтной базы; во-вторых, пытаться представить конфликтные базы в некоторой компактной форме, требующей существенно меньшего объема памяти для хранения. Оба данных решения имеют свои отрицательные стороны:

1. Любая известная техника удаления выученных дизъюнктов является по своей природе эвристикой и не дает никакой гарантии того, что удаленный выученный дизъюнкт не будет вновь порожден в дальнейшем;
2. Работа с конфликтными базами, представленными не в виде множеств дизъюнктов, а в виде некоторых более компактных структур, может быть слишком медленной, что опять-таки приведет к снижению эффективности поиска.

Решению обеих описанных выше проблем посвящена настоящая диссертация. Конкретно, для решения первой проблемы были предложены эвристические техники определения полезности выученного дизъюнкта на основе числа его порождений/удалений в процессе поиска: чем больше повторных порождений имеет некоторый дизъюнкт, тем он считается более полезным для процесса вывода. Этот подход является, по-видимому, новым, так как похожих идей в известных источниках найти не удалось. Тем не менее его практическая значимость может аргументироваться тем фактом, что SAT-решатель MapleLCMDistChronoBT-DL-v3, в котором были реализованы алгоритмы отслеживания и использования повторно порождаемых выученных дизъюнктов, победил на соревнованиях SAT-решателей SAT Race 2019[7].

Что касается второй проблемы, то, исходя из работ Ф. Чаталика, Ф. Алула, С. Гопалакришнана, А.А. Семёнова, И.В. Отпущенникова и др., для ее решения могут быть использованы специальные классы решающих диаграмм (decision diagrams): двоичные диаграммы решений и их редуцированные ва-

[7]https://satcompetition.github.io/2019/results.html.

рианты (Reduced Ordered Binary Decision Diagrams)[8, 9]; двоичные диаграммы с подавлением нуля (Zero-Suppressed Binary Decision Diagrams, ZDD)[10], а также дизъюнктивные диаграммы (DJD), введенные в работе А.А. Семёнова и И.В. Отпущенникова[11]. Следует отметить, что, к сожалению, перечисленные структуры являются либо в принципе неэффективными в худшем случае (ROBDD), либо довольно медленными (ZDD, DJD), что ведет к высоким накладным расходам при их использовании. С другой стороны, в настоящей диссертации и сопутствующих ей работах показано, что дизъюнктивные диаграммы позволяют эффективно извлекать из исходной формулы либо из конфликтной базы новую информацию, учет которой ускоряет работу SAT-решателя на исходной формуле. Иногда такое ускорение может быть весьма существенным, и это особенно ценно для экстремально трудных формул, возникающих в задачах проверки эквивалентности булевых схем.

Таким образом, резюмируя все сказанное выше, можем заключить, что настоящая диссертация посвящена актуальной проблеме разработки алгоритмов и техник работы с конфликтными ограничениями, порождаемыми современными SAT-решателями, основанными на стратегии CDCL. Как будет продемонстрировано далее, разработанные техники в ряде случаев дают весьма существенное снижение времени работы SAT-решателей на основе алгоритма CDCL на аргументированно трудных промышленных тестах.

Цель и задачи исследования. Основная цель состоит в повышении эффективности современных SAT-решателей, основанных на алгоритме CDCL, при помощи новых алгоритмов генерации и обработки конфликтных ограничений.

Для достижения поставленной цели были решены следующие задачи:

- изучен феномен повторного порождения конфликтных ограничений в современных SAT-решателях, основанных на алгоритме CDCL;
- разработаны алгоритмы эффективного отслеживания повторно порождаемых конфликтных ограничений;
- на основе разработанных алгоритмов мониторинга повторно порождаемых конфликтных ограничений разработаны новые стратегии, дополняющие базовый алгоритм CDCL;
- на базе разработанных стратегий реализованы последовательный и па-

[8]Lee, C.Y. Representation of switching circuits by binary-decision programs / C.Y. Lee // The Bell System Technical Journal. — 1959. — Vol. 38, no. 4. — P. 985–999.

[9]Bryant, R.E. Graph-Based Algorithms for Boolean Function Manipulation / R.E. Bryant // IEEE Trans. Computers. — 1986. — Vol. 35, no. 8. — P. 677–691.

[10]Minato, S.-I. Zero-Suppressed BDDs for Set Manipulation in Combinatorial Problems / S.-I. Minato // Proceedings of DAC '93. — ACM, 1993. — P. 272–277.

[11]Semenov, A. On one class of decision diagrams / A. Semenov, I. Otpuschennikov // Automation and Remote Control. — 2016. — Vol. 77, no. 4. — P. 617–628.

раллельный SAT-решатели, превосходящие известные SAT-решатели по эффективности на некоторых классах трудных комбинаторных задач в форме SAT;
- в рамках методов преобразования конфликтных баз, генерируемых CDCL SAT-решателем, разработан эффективный алгоритм перенаправления путей в дизъюнктивной диаграмме;
- разработан алгоритм порождения новых конфликтных ограничений на основе дизъюнктивной диаграммы, построенной по исходной формуле в КНФ;
- проведены вычислительные эксперименты, подтверждающие практическую эффективность разработанных алгоритмов.

Методы и инструменты исследования. Теоретическая часть исследования основана на методах дискретной математики, теории вычислительной сложности, вычислительной логике, методах параллельных вычислений, теории булевых функций, а также на анализе известных результатов, относящихся к обработке массивов конфликтных ограничений при решении проблемы булевой выполнимости.

Для построения по алгоритмическим описаниям дискретных функций формул и специальных графов (And-Inverter Graphs) применялся программный комплекс Transalg, разработанный в ИДСТУ СО РАН[12, 13]. В качестве платформ для реализации разработанных методов, а также для проведения вычислительных экспериментов использовались современные SAT-решатели (MiniSat2.2, MapleLCMDist, Painless, Kissat и др.). При работе с дизъюнктивными диаграммами в роли SAT-оракула использовался программный комплекс PySAT[14]. Большая часть вычислительных экспериментов проведена на кластере ИДСТУ СО РАН “Академик В.М. Матросов”[15].

Научная новизна. Новыми являются все основные результаты, полученные и использованные в диссертации, в том числе:

1. Алгоритмы и техники, отслеживающие и использующие повторно порождаемые конфликтные ограничения для ускорения базового алгоритма CDCL;
2. Программный комплекс RADAR, использующий SAT-решатель Minisat в комбинации с алгоритмами отслеживания повторно порождаемых

[12]Encoding Cryptographic Functions to SAT Using TRANSALG System / I. Otpuschennikov, A. Semenov, I. Gribanova et al. // ECAI. — 2016. — P. 1594–1595.

[13]Translation of Algorithmic Descriptions of Discrete Functions to SAT with Applications to Cryptanalysis Problems / A. Semenov, I. Otpuschennikov, I. Gribanova et al. // Logical Methods in Computer Science. — 2020. — Vol. 16, no. 1. — P. 1–42.

[14]Ignatiev, A. PySAT: A Python Toolkit for Prototyping with SAT Oracles / A. Ignatiev, A. Morgado, J.P. Marques-Silva // SAT. — 2018. — P. 428–437.

[15]Irkutsk Supercomputer Center of SB RAS. — 2023. — URL: http://hpc.icc.ru.

конфликтных ограничений, разработанный с использованием стандарта MPI;
3. Алгоритм разбиения трудных формул, используемый для решения задач проверки эквивалентности булевых схем;
4. Полиномиальный алгоритм перенаправления путей в дизъюнктивной диаграмме, сохраняющий структуру диаграммы;
5. Многопоточный программный комплекс для работы с дизъюнктивными диаграммами PyDJD, использующий SAT-оракул PySAT и алгоритм перенаправления путей для генерации новых конфликтных ограничений.

Положения, выносимые на защиту:

1. Алгоритм отслеживания повторно порождаемых конфликтных ограничений SAT-решателями, основанными на стратегии CDCL;
2. Параллельный модульный SAT-решатель RADAR[16], основанный на алгоритме отслеживания повторно порождаемых конфликтных ограничений;
3. Семейство алгоритмов для эффективной работы с дизъюнктивными диаграммами;
4. Программный комплекс PyDJD[17], реализующий алгоритмы работы с дизъюнктивными диаграммами с целью порождения новых конфликтных ограничений в трудных примерах SAT.

Теоретическая и практическая значимость. Полученные результаты представляют теоретическую и практическую важность в силу того, что разработанные методы и алгоритмы работы с конфликтными ограничениями могут быть использованы для ускорения решения комбинаторных задач из широкого класса, сведенных к проблеме булевой выполнимости (SAT).

Соответствие специальности. Научно-квалификационная работа охватывает такие направления, как разработка, реализация и тестирование эффективных численных методов решения различных классов SAT-задач, включающих верификацию аппаратного обеспечения и криптоанализ. Разработанные алгоритмы реализованы в виде однопоточных и параллельных программных комплексов. На основе вышесказанного можно утверждать, что данная работа соответствует специальности 1.2.2 “Математическое моделирование, численные методы и комплексы программ”. Полученные в работе результаты соответствуют паспорту специальности 1.2.2 в пунктах 2, 3, 9.

Достоверность результатов проведенных исследований. Достоверность полученных в работе теоретических результатов обеспечивается коррект-

[16]https://github.com/Vikseko/RadarSatSolver.

[17]https://github.com/Vikseko/PyDJD.

ностью предложенных алгоритмов, эффективность которых обоснована как теоретически, так и экспериментальным путем.

Апробация результатов работы. Результаты, представленные в диссертации, докладывались и обсуждались на следующих конференциях: “56-я Международная научная студенческая конференция (МНСК)” (г. Новосибирск, 2018 г.), “Винеровские чтения” (г. Иркутск, 2018 г., 2019 г.), “ECAI 2020: 24th European Conference on Artificial Intelligence” (Испания, г. Сантьяго-де-Компостела, 2020 г.), “MIPRO 2020: 43rd International Convention on Information, Communication and Electronic Technology” (Хорватия, г. Опатия, 2020 г.), “Параллельные вычислительные технологии (ПаВТ 2022)” (г. Дубна, 2022 г.), “MIPRO 2022: 45th Jubilee International Convention on Information, Communication and Electronic Technology” (Хорватия, г. Опатия, 2022 г.), “Ляпуновские чтения” (г. Иркутск, 2018–2022 гг.,), “Параллельные вычислительные технологии (ПаВТ 2023)” (г. Санкт-Петербург, 2023 г.), “MIPRO 2023: 46th International Convention on Information, Communication and Electronic Technology” (Хорватия, г. Опатия, 2023 г.).

Основные результаты по теме исследования получены в рамках следующих проектов:

- грант РНФ № 16-11-10046 “Применение параллельных и распределенных алгоритмов решения проблемы булевой выполнимости (SAT) к криптоанализу, поиску комбинаторных структур и исследованию дискретных моделей коллективного поведения”;
- грант РНФ № 22-21-00583 “Методы оценивания декомпозиционной трудности булевых формул с применением к задачам синтеза и верификации дискретных управляющих систем”.

Публикации и личный вклад автора. Результаты диссертации опубликованы в 12 печатных и 4 электронных работах, из которых одна статья опубликована в рецензируемом журнале, входящем в перечень ВАК, четыре статьи опубликованы и две приняты к печати в изданиях, индексируемых в международных системах научного цитирования (Web of Science, Scopus).

Базовый алгоритм отслеживания и использования повторно порождаемых конфликтных ограничений в рамках CDCL-вывода предложен в неделимом соавторстве с О.С. Заикиным, С.Е. Кочемазовым и А.А. Семёновым. Алгоритм оценивания декомпозиционной трудности формул, кодирующих эквивалентность булевых схем, был разработан авторским коллективом в составе А.А. Семёнова, К.И. Чухарева, Е.А. Тарасова, Д.С. Чивилихина, В.С. Кондратьева. Версии алгоритма, использующего повторно порождаемые выученные дизъюнкты, реализованные в портфолио-решателе RADAR и в последовательных решателях MapleLCMDist-mod и MapleLCMDistChronoBT-mod, разрабо-

таны В.С. Кондратьевым. Алгоритм перенаправления путей в дизъюнктивной диаграмме и алгоритмы порождения конфликтных ограничений за счет применения SAT оракула к дизъюнктивной диаграмме предложены и программно реализованы В.С. Кондратьевым. Таким образом, на защиту выносятся результаты, полученные лично автором.

Благодарности. Автор выражает глубокую признательность научному руководителю к.т.н. А.А. Семёнову за руководство научно-исследовательской работой, к.т.н. И.В. Отпущенникову, к.т.н. О.С. Заикину, С.Е. Кочемазову, к.ф.-м.н. А.С. Игнатьеву и И.А. Грибановой за полезные замечания в ходе обсуждения результатов работы.

КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

В **главе 1** формулируются базовые понятия, дается описание алгоритмов, которые используются при решении задачи о булевой выполнимости (SAT).

В **п. 1.1** формулируются задача удовлетворения ограничений (CSP) и задача о булевой выполнимости (SAT). Приводятся примеры формулировок комбинаторных задач как в виде CSP, так и в форме SAT.

В **п. 1.2** вводятся понятия дискретной функции, булевой формулы, булевой схемы. Формулируется задача проверки эквивалентности булевых схем (LEC). Описывается сведение LEC к SAT при помощи преобразований Цейтина[18].

В **п. 1.3** в контексте исторической ретроспективы описаны несколько алгоритмов решения SAT, не являющихся предметом рассмотрения данной диссертации.

В **п. 1.4** приводится описание работы алгоритма CDCL. Ключевой особенностью данного алгоритма является способность запоминать информацию о ходе поиска и использовать ее для повышения эффективности решения SAT. Формулируются понятия “выученный дизъюнкт”, “граф вывода”, “уровни решения”, “нехронологический бектрекинг”, “рестарт”, “база конфликтных ограничений” и “эвристики выбора переменных”. Приводится пример работы алгоритма CDCL.

В **п. 1.5** описывается техника работы с выученными дизъюнктами, которая заключается в нахождении новых дизъюнктов, “поглощающих” исходные. Данная техника чаще всего фигурирует в работах под названием “минимизация выученных дизъюнктов” (LCM)[19].

---

[18] Tseitin, G.S. On the Complexity of Derivation in Propositional Calculus / G.S. Tseitin // Studies in Constructive Mathematics and Mathematical Logic. Part II. — 1970. — P. 115–125.

[19] An Effective Learnt Clause Minimization Approach for CDCL SAT Solvers / M. Luo, C.-M. Li, F. Xiao et al. // International Joint Conference on Artificial Intelligence (IJCAI). — 2017. — P. 703–711.

В **п. 1.6** вводятся диаграммы решений как способ компактного представления массивов выученных дизъюнктов. Описываются три типа диаграмм решений: сокращенные упорядоченные двоичные диаграммы решений (ROBDD), двоичные диаграммы решений с подавлением нуля (ZDD) и дизъюнктивные диаграммы (DJD), разработанные А.А. Семёновым и И.В. Отпущенниковым специально для компактного представления баз конфликтных ограничений, порождаемых SAT-решателями на базе алгоритма CDCL.

В **главе 2** приводятся теоретические основы разработанных в ходе исследования алгоритмов, связанных с генерацией и обработкой конфликтных ограничений.

В **п. 2.1** описываются последовательная и параллельная версии алгоритма, использующего повторно порождаемые конфликтные ограничения для ускорения решения SAT. Разработанные алгоритмы основаны на описанной ниже концепции.

Пусть A — полный алгоритм решения SAT, основанный на CDCL (можно считать таковым любой CDCL SAT-решатель), и пусть C — произвольная КНФ. Предположим, что C сравнительно трудна для A и A при работе с C совершает некоторое количество рестартов, выполняя перед каждым последующим рестартом чистку конфликтной базы, хранящейся в виде хеш-таблицы $H(C,A)$. Полагаем, что доступ к $H(C,A)$ происходит за время $O(1)$ (ввиду прямого доступа к памяти). Первым рестартом будем считать собственно старт алгоритма A. После завершения первого рестарта происходит чистка конфликтной базы. Все выученные дизъюнкты, удаляемые из конфликтной базы, сохраняются в хеш-таблицу $H(C,A)$. После завершения рестарта с номером k, $k \geq 2$, каждый выученный дизъюнкт проверяется на предмет его встречи ранее (за счет проверки хеш-таблицы), и число встреч обновляется в $H(C,A)$. После того как дизъюнкт встретился некоторое заданное число раз d, принимается решение о его неудалении в дальнейшем.

Данная идея была реализована в двух работах[20, 21], результаты соответствующих экспериментов приведены в главе 3 настоящей диссертации.

В **п. 2.2** приводится описание весьма общих техник[22], позволяющих строить верхние оценки трудности трудных вариантов SAT относительно полных SAT-решателей.

---

[20]Кондратьев, В.С. Дубликаты конфликтных ограничений в CDCL-выводе и их использование в задачах обращения некоторых криптографических функций / В.С. Кондратьев, А.А. Семенов, О.С. Заикин // Вычислительные методы и программирование. — 2019. — Т. 20. — С. 54–66.

[21]Кондратьев, В.С. Применение дубликатов конфликтных ограничений для ускорения работы алгоритмов решения проблемы булевой выполнимости / В.С. Кондратьев // Винеровские чтения: Материалы Всерос. молодежной науч.-практ. конф. Иркутск, 06–07 июня 2019 г. — Иркутск: ИРНИТУ, 2019. — С. 19–25.

[22]Evaluating the Hardness of SAT Instances Using Evolutionary Optimization Algorithms / A. Semenov, D. Chivilikhin, A. Pavlenko et al. // CP. — 2021. — Vol. 210. — P. 47:1–47:18.

В **п. 2.2.1** вводятся понятия лазейки, сильной лазейки, а также ряд понятий, связанных с оценками трудности. На основании данных понятий и понятия "разбиение SAT"[23] формулируется понятие декомпозиционной трудности формулы C относительно полного алгоритма решения SAT A и разбиения SAT $\Pi = \{G_1, \ldots, G_s\}$, представленное следующей величиной:

$$\mu_{A,\Pi}(C) = \sum_{j=1}^{s} t_A(G_i \wedge C),$$

где через $t_A(C)$ обозначено время работы алгоритма A на формуле (КНФ) C. Мы можем связать с произвольным разбиением SAT случайную величину ξ_Π, значения которой равны времени работы SAT-решателя A на формулах вида $G \wedge C, G \in \Pi$. Справедливо следующее соотношение:

$$\mu_{A,\Pi}(C) = |\Pi| \cdot E[\xi_\Pi].$$

Для оценки $E[\xi_\Pi]$ можно использовать величину $1/N \cdot \sum_{j=1}^{N} \xi_j$ в соответствии с методом Монте–Карло[24]. В этом случае ξ_j, $j \in \{1, \ldots, N\}$, — одно наблюдение случайной величины $\xi_\Pi : \xi_j = t_A(G \wedge C)$, где G — формула, выбранная из Π в соответствии с заданным на Π равномерным распределением.

В **п. 2.2.2** описана конструкция, применяемая для оценивания трудности КНФ, кодирующих задачи проверки логической эквивалентности булевых схем (LEC). Для построения оценки декомпозиционной трудности КНФ C (кодирующей задачу эквивалентности двух булевых схем), в которой переменные из множества $X^{\text{in}} = \{x_1, x_2, \ldots, x_n\}$ кодируют вход схем, требуется разделить множество X^{in} на непересекающиеся подмножества по k переменных (при $k = 2$: $\{x_1, x_2\}, \{x_3, x_4\}, \ldots$). Для каждого из таких подмножеств, называемых чанками (chunk), строится сбалансированная булева функция (функция, принимающая значения 1 и 0 на равном числе значений входов), а также функция, являющаяся ее отрицанием. Эти функции связывают переменные, находящиеся в рассматриваемом множестве (например, при $k = 2$: $\phi = x_i \oplus x_{i+1}$ либо $\phi = \neg(x_i \oplus x_{i+1})$). Далее можно сформировать случайную задачу, выбрав для каждого чанка случайным образом связанную с ним сбалансированную функцию либо ее отрицание. Соответственно, всего имеем $2^{\lceil n/k \rceil}$ задач вида $C \wedge \phi_1 \wedge \phi_2 \wedge \cdots \wedge \phi_{\lceil n/k \rceil}$. Для оценки общего времени решения всех таких задач используем метод Монте–Карло, оценивая $E[\xi_\Pi]$ по случайной выборке.

[23] Hyvärinen, A.E.J. Grid based propositional satisfiability solving: Ph.D. thesis / Antti E. J. Hyvärinen. — Aalto University, Helsinki, Finland. — 2011.

[24] Metropolis, N. The Monte Carlo method / N. Metropolis, S. Ulam // Journal of the American statistical association. — 1949. — Vol. 44, no. 247. — P. 335–341.

Исходная формула невыполнима тогда и только тогда, когда невыполнимы все формулы из данного разбиения.

В **п. 2.2.3** описывается алгоритм проведения ATPG-тестирования по модели одиночной неисправности ("stuck-at fault"). В частности, описывается процесс проведения полного ATPG-тестирования булевых схем при условии одновременной поломки только одного "гейта" (узла, вентиля) схемы. Идея разработанного алгоритма заключается в том, чтобы итеративно решать ATPG-задачи с ограничением по времени. На первом шаге устанавливается низкое ограничение по времени решения, запускается параллельное решение всех ATPG-подзадач для рассматриваемой булевой схемы, построенных по модели одиночной неисправности. На следующем шаге граница максимального времени решения устанавливается выше, и решение нерешенных на предыдущем шаге задач запускается вновь. Данный алгоритм назван *k-шаговая SAT-фильтрация R*. В результате выполнения алгоритма остаются только наиболее трудные ATPG-задачи, которые можно решить, используя метод построения разбиения SAT, описанный в п. 2.2.2.

Пункт **2.3** главы 2 сконцентрирован на новых методах работы с дизъюнктивными диаграммами. Описан разработанный алгоритм перенаправления путей в дизъюнктивной диаграмме из одной терминальной вершины в другую с сохранением структуры диаграммы. Данный алгоритм может использоваться как часть основанного на дизъюнктивных диаграммах препроцессинга в применении к трудным формулам в КНФ.

В **п. 2.3.1** описывается способ получения дополнительной информации из дизъюнктивной диаграммы при помощи SAT-оракула. Это становится возможным благодаря тому, что каждый путь в дизъюнктивной диаграмме в ?-вершину можно представить в виде набора литералов и проверить совместность данного набора с исходной формулой. Если исходная формула выполнима, то найдется совместный с ней путь в ?-вершину (утверждение 2). В свою очередь, для каждого пути в ?-вершину, для которого доказана несовместность с исходной формулой, возможно построить дизъюнкт, запрещающий сочетание значений переменных, соответствующее этому пути. При добавлении такого дизъюнкта к формуле получается КНФ, эквивалентная исходной (утверждение 3).

В **п. 2.3.2** описывается алгоритм перестраивания диаграммы с учетом результатов проверки путей в ?-вершину. Каждый путь, для которого SAT-оракул доказал несовместность с исходной формулой, можно перенаправить в диаграмме из ?-вершины в 1-вершину с сохранением структуры диаграммы. Для этого необходимо понять, пересекается ли этот путь с какими-либо другими. Если нет, то достаточно удалить связь последней вершины пути с ?-вершиной и добавить связь той же полярности с 1-вершиной. В случае же, если путь пересекается с

другими в каких-либо своих вершинах, требуется выделить часть пути от вершины, ближайшей к ?-вершине, до первой вершины, у которой больше, чем один родитель. Создав копии всех вершин выделенной части (копии вершин связаны с теми же потомками, что и оригиналы) и перенаправив путь из оригинальных вершин в копии, мы получаем возможность перенаправить конкретный путь из ?-вершины в 1-вершину, не затрагивая никаких других путей и не нарушая структуру диаграммы. В этом и состоит суть предложенного в данном пункте перенаправления путей. Показано, что задача перенаправления конкретного пути решается за полиномиальное от числа вершин в диаграмме время.

В **главе 3** диссертации описаны результаты вычислительных экспериментов, проведенных с использованием методов, представленных в главе 2. В силу того, что тема исследования достаточно обширна, результаты разделены на 3 раздела, эксперименты в каждом из которых соответствуют различным техникам работы с конфликтными ограничениями.

В экспериментальной части данного исследования рассматривались несколько классов задач, вычислительные эксперименты проводились на различных платформах. В **п. 3.1** приводится описание как классов тестовых задач, так и конфигурации вычислительных платформ.

В **п. 3.1.1** описаны эксперименты с криптографическими хеш-функциями. Конкретно, рассмотрены следующие задачи: для полнораундовой хеш-функции MD4 найти такое произвольное 512-битное сообщение, хеш которого содержит k нулевых старших бит ($k = 20, 21, \ldots, 27$); задача обращения неполнораундовых версий хеш-функции SHA-1; поиск второго блока двухблоковой коллизии для хеш-функции MD5 при разностных соотношениях из работы С. Ван и Х. Ю[25]. Помимо этого рассматривались примеры SAT, которые кодируют криптоанализ ряда генераторов ключевого потока, а также блочного шифра DES.

В **п. 3.1.2** рассматриваются особо трудные экземпляры LEC-задач для схем, реализующих некоторые арифметические функции, такие как умножители и алгоритмы сортировки. В качестве рассматриваемых алгоритмов умножения были выбраны: “алгоритм умножения столбиком” (Column), “дерево Уоллеса” (Wallace), “разложение Карацубы” (Karatsuba) и “умножитель Дадда” (Dadda). В качестве рассматриваемых алгоритмов сортировки были выбраны: пузырьковая сортировка (Bubble), сортировка выбором (Selection) и “блинная” сортировка (Pancake).

Еще одним “классом” задач в данном исследовании (**п. 3.1.3**) были большие наборы разнородных тестов, использованных на ежегодных соревновани-

[25]Wang, X. How to Break MD5 and Other Hash Functions / X. Wang, H. Yu // Advances in Cryptology — EUROCRYPT 2005: Proc. of 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, 22–26 May 2005 / Ed. by R. Cramer. — Springer, 2005. — Vol. 3494. — P. 19–35.

Таблица 1 — Результаты применения различных решателей к обращению MD4-хешей с k нулевыми старшими битами. Для каждого значения k указано среднее время в секундах по 10 независимым запускам

	Число k старших нулевых бит MD4-хеша							
Решатель	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**
RADAR	600	725	1161	1498	2155	3730	2918	22284
Painless	4724	7401	11262	27223	66078	135097	127941	75167
Plingeling	9054	2527	11846	12573	4444	20807	52793	>200000

ях алгоритмов решения задачи булевой выполнимости в 2017, 2018, 2019 и 2021 годах. Описание всех тестов можно найти на официальном сайте SAT Competitions[26].

В **п. 3.1.4** описаны две вычислительные платформы, на которых в рамках исследования проводились эксперименты: вычислительный кластер "Академик В.М. Матросов" ИДСТУ СО РАН, использующийся в большинстве экспериментов, и вычислительный узел Yandex Cloud, базирующийся на платформе Intel Ice Lake.

В **п. 3.2** описываются разработанные SAT-решатели, в основе которых лежит эффект ускорения работы решателя за счет использования повторно порождаемых конфликтных ограничений. В **п. 3.2.1** представлен прототип параллельного модульного SAT-решателя RADAR и эксперименты по сравнению его эффективности с актуальными параллельными SAT-решателями (Plingeling, Treengeling, Painless-mapleCOMSPS) на задачах, перечисленных в п. 3.1.1. Все эти решатели являются рандомизированными. Поэтому в каждом конкретном эксперименте рассматривалось по 10 копий одного теста. Далее время работы по этим 10 тестам усреднялось, и полученное значение рассматривалось как численный результат эксперимента. Время решения одной копии задачи ограничивалось 200 000 секунд. В каждом эксперименте данного раздела в роли вычислительной платформы использовался один узел кластера (см. п. 3.1.4). В таблице 1 приведены результаты применения рассматриваемых решателей к задачам обращения полнораундовой хеш-функции MD4 с разным числом подставленных нулевых старших бит. Дополнительно отметим, что при $k = 24$ Painless решил 8 задач из 10, при $k = 26$ Painless решил 7 задач из 10, при $k = 27$ этим решателем были решены только 3 задачи из 10. Решатель Plingeling при $k = 27$ не решил ни одной задачи. Решатель Treengeling не справился ни с одной из копий рассматриваемых задач за отведенное время. В свою очередь, RADAR решил все предложенные задачи. В каждом экспери-

[26] http://www.satcompetition.org.

Таблица 2 — Результаты решений задачи обращения случайных значений хеш-функции SHA-1-22 и задачи поиска вторых блоков для двухблоковой коллизии MD5. Для каждого теста указано среднее время в секундах по 10 независимым запускам

	SHA-1-22					**MD5**	
Решатель	**Тест № 1**	**Тест № 2**	**Тест № 3**	**Тест № 4**	**Тест № 5**	**Тест № 1**	**Тест № 2**
RADAR	812	1424	1218	1718	1235	734	821
Painless	1346	1852	922	936	1205	18318	4742
Plingeling	30702	58890	18085	93547	3384	32763	11620
Treengeling	36605	118195	147424	44439	>200000	26430	25909

менте среднее время вычислялось только по тем задачам, на решение которых потребовалось не более 200 000 секунд.

Таблица 2 содержит результаты применения SAT-решателей к задаче обращения 22-раундовой версии хеш-функции SHA-1 (SHA-1-22) и к задаче поиска вторых блоков для двухблоковой коллизии MD5. Были сгенерированы 5 случайных хеш-значений данной функции, после чего для каждого хеш-значения вычисления запускались для 10 копий задачи обращения этого значения. Стоит отдельно отметить, что решатель Treengeling не дорешал ни одной копии теста № 5 для задачи SHA-1-22 за отведенное время.

По результатам экспериментов можно сделать вывод, что на SAT-задачах, кодирующих проблемы обращения криптографических хеш-функций, решатель RADAR продемонстрировал существенное превосходство над многопоточными решателями, занимавшими высокие места на специализированных соревнованиях.

В **п. 3.2.1** описываются два SAT-решателя на основе известных решателей MapleLCMDist и MapleLCMDistChronoBT, включающие расширенный вариант эвристики, основанной на работе с повторно порождаемыми конфликтными ограничениями:

В течение некоторого времени, ограниченного числом рестартов, происходит анализ базы конфликтных ограничений и добавление всех полученных ограничений в хеш-таблицу. В случае если ограничение уже содержится в хеш-таблице, оно добавляется напрямую в уровень Core (согласно иерархии выученных дизъюнктов в SAT-решателях на основе COMINISATPS[27]*, описанной в п. 2.1).*

Результаты работы построенных решателей сравнивались с их немодифицированными версиями на различных тестах, взятых с соревнований алго-

[27]Oh, C. Between SAT and UNSAT: The Fundamental Difference in CDCL SAT / C. Oh // Theory and Applications of SAT Testing — SAT 2015 / Ed. by M. Heule et al. — Springer Int. Publ., 2015. — P. 307–323.

Таблица 3 — Статистика количества решенных задач на тестах SAT Competition 2017 и 2018, задачи решались с ограничением по времени решения в 5000 секунд

	SC 2017			SC 2018		
Решатель	**Решено**	**SAT**	**UNSAT**	**Решено**	**SAT**	**UNSAT**
MapleLCMDist	206	100	106	228	128	100
MapleLCMDist-mod	206	100	106	237	136	101
MapleLCMDistChronoBT	212	98	114	234	132	102
MapleLCMDistChronoBT-mod	211	100	111	239	136	103

ритмов для решения задачи булевой выполнимости, проводимых в 2017 и 2018 годах. Полученные результаты приведены в таблице 3. Таблица 3 демонстрирует для каждого решателя: в столбцах “Решено” — общее число решенных задач, в столбцах “SAT” — число решенных выполнимых задач, в столбцах “UNSAT” — число решенных невыполнимых задач. Как можно понять из таблицы 3, построенные решатели MapleLCMDist-mod и MapleLCMDistChronoBT-mod показали сопоставимые результаты с оригинальными немодифицированными решателями на тестах с SAT Competition 2017, но существенно выиграли у них на задачах с SAT Competition 2018.

Коллективом, в который входил автор данного исследования, была разработана еще одна, более эффективная версия решателя, использующего эвристику повторно порождаемых конфликтных ограничений, получившая название MapleLCMDistChronoBT-DL-v3[28]. Данная версия отличается от $*$-mod версий в том числе тем, что работа с хеш-таблицей, хранящей конфликтные ограничения, не завершается после первых рестартов, а продолжается до момента завершения алгоритма. Для того чтобы избежать переполнения хеш-таблицы, реализована процедура ее периодической чистки. Решатель MapleLCMDistChronoBT-DL-v3 участвовал в соревнованиях SAT Race 2019, где занял первое место в двух категориях из трех основных по системе начисления очков PAR 2 и второе место по числу решенных выполнимых задач[29].

В **п. 3.3** описываются алгоритмы работы с конфликтной информацией, ускоряющие решение задач, относящихся к такой важной области как “Проблема выполнимости булевых схем” (Circuit SAT).

В частности, в **п. 3.3.1** описываются эксперименты по построению оценок декомпозиционной трудности КНФ, кодирующих задачи эквивалентности схем-

[28]Speeding Up CDCL Inference with Duplicate Learnt Clauses / S. Kochemazov, O. Zaikin, A. Semenov, V. Kondratiev // Proc. of ECAI 2020: 24th European Conference on Artificial Intelligence. Santiago de Compostela, Spain, 29 August – 8 September 2020 / Ed. by G. De Giacomo, A. Catala, B. Dilkina et al. — IOS Press, 2020. — Vol. 325. — P. 339–346.

[29]https://satcompetition.github.io/2019/results.html.

ных реализаций алгоритмов умножения и сортировки. В качестве основного инструмента в данной серии экспериментов использовалась техника оценивания декомпозиционной трудности LEC задач, описанная в разделе 2.2.2. Результаты экспериментов показывают, что предложенный метод разбиения SAT дает очень низкую дисперсию. Это существенное преимущество предложенного метода, так как он позволяет использовать небольшую случайную выборку для получения надежной оценки $E[\xi_{\Pi}]$.

В **п. 3.3.2** демонстрируются результаты вычислительных экспериментов по проведению ATPG-тестирования в отношении модели ошибки "stuck-at fault" с использованием алгоритмов, описанных в разделах 2.2.2 и 2.2.3. В качестве тестовых задач рассматривались схемы, представляющие алгоритмы, которые перемножают два Q-битных числа, где $Q \in \{20, 32\}$. Далее используются обозначения C_Q, W_Q, D_Q, K_Q для алгоритмов Column, Wallace, Dadda и Karatsuba соответственно (см. п. 3.1.2). Количество подзадач в полном тестовом наборе ATPG для каждого из рассматриваемых алгоритмов равно удвоенному количеству узлов (гейтов, вентилей) соответствующей схемы, так как каждый узел может "застрять" либо в 0, либо в 1.

На каждом шаге фильтрации алгоритм пытался решить все подзадачи, оставшиеся нерешенными с предыдущего шага, но с большим лимитом времени. Так, для задач с $Q = 20$ лимит времени на первом шаге составлял 2 секунды, на втором шаге он был увеличен до 100 секунд, а на третьем — до 43200 секунд или 12 часов. Для $Q = 32$, в связи с тем, что соответствующие задачи значительно сложнее, лимит времени был увеличен для первых двух шагов: до 3 секунд на первом шаге и до 1000 секунд на втором. Все задачи ATPG для C_Q, W_Q, D_Q, K_Q, $Q \in \{20, 32\}$, оказались легкими для Kissat (самое большое время решения составило 2353 секунд для одной из задач для C_{32}). Однако ряд тестов для алгоритмов K_{20} и K_{32} оказались чрезвычайно трудными: Kissat последовательно не смог решить их за 12 часов. Тем не менее, все такие задачи для K_{20} были решены параллельно с использованием техники, описанной в разделе 2.2.2.

Результаты данной части эксперимента представлены в таблице 4. Во всех столбцах таблицы 4, касающихся времени, оно представлено в секундах. Столбец "Задача" содержит имя ATPG-задачи для K_{20}, которое включает номер, идентифицирующий "stuck-at fault" узел и его полярность ("n" для отрицательных и "p" для положительных). Столбец "Число подзадач" показывает общее количество подзадач в разбиении. В столбце "Среднее время" приведено среднее значение времени, необходимого для решения одной подзадачи из разбиения, а в столбце "Общее время" — общее время, необходимое для решения всех подзадач с использованием одного ядра процессора. Столбец "Реальное время (96

Таблица 4 — Решение трудных ATPG-задач для K_{20} с использованием разбиения SAT по переменным входа

Задача	**Число подзадач**	**Среднее время**	**Общее время**	**Реальное время (96 ядер)**
19187_n	1024	460.24	471285.76	5618
19188_n	1024	450.17	460974.08	5494
19189_p	1024	450	460800.00	5493
19192_p	1024	1681.61	1721968.64	19670
19193_n	1024	1685.78	1726238.72	19794
19648_n	1024	911.63	933509.12	10501
19649_p	1024	910.97	932833.28	10480
19656_n	1024	1399.76	1433354.24	16261
19657_n	1024	1402.14	1435791.36	16329

ядер)" показывает время, которое потребовалось для решения всех подзадач из соответствующего разбиения: оно соответствует общему времени, которое потребуется пользователю для решения LEC-задачи с использованием данного разбиения на 96 вычислительных ядрах описанной в п. 3.1.4 конфигурации платформы Yandex Cloud.

К сожалению, аналогичные 22 задачи для K_{32}, которые не были решены алгоритмом 3-шаговой фильтрации, оказались весьма трудными даже для решения с помощью описанной стратегии разбиения, и в этом случае мы имеем только их оценки трудности, которые очень велики. Так, например, для одной из таких задач, согласно построенным оценкам, общее время полного решения подзадач из декомпозиции составит не менее $1.54 \cdot 10^8$ секунд (почти 20 дней работы на 96 ядрах используемой конфигурации).

Таким образом, можно заключить, что предложенный метод позволил для всех рассмотренных алгоритмов умножения, за исключением K_{32}, построить полные системы тестов ATPG, выявляющие одиночную неисправность указанного вида для каждого узла соответствующей схемы.

В **п. 3.4** описывается использование дизъюнктивных диаграмм для обработки конфликтной информации и ускорения решения трудных вариантов SAT. Напомним, что дизъюнктивные диаграммы первоначально предназначались для компактного представления конфликтных баз, порождаемых современными SAT-решателями.

В **п. 3.4.1** описываются результаты экспериментов по сравнению эффективности на данной задаче дизъюнктивных диаграмм с диаграммами с подавлением нуля (ZDD), которые также могут быть использованы для этих целей.

В вычислительном эксперименте использовались несколько программных комплексов: программный комплекс DDIAGRAMS, предназначенный для построения DJD (автор — И.В. Отпущенников); программный комплекс ZDDGEN, который строит ZDD-представления произвольных ДНФ, рассматривая их как наборы подмножеств литералов; SAT-решатель ZRES (автор — Ф. Чаталик), который использует ZDD для представления CNF. В качестве тестовых задач использовались экземпляры SAT, которые кодируют криптоанализ нескольких генераторов ключевого потока и SAT-кодирование блочного шифра DES (см. п. 3.1.1). Результаты экспериментов продемонстрировали, что DJD лучше ZDD в большинстве тестов по количеству вершин. Также DDIAGRAMS проигрывает ZDDGEN и ZRES по времени построения диаграммы только на одном тесте (DES_56/64).

В **п. 3.4.2** приведены вычислительные эксперименты по использованию дизъюнктивных диаграмм для препроцессинга произвольных КНФ. В экспериментах рассматриваемые КНФ подавались на вход программе PyDJD[30], в которой реализованы алгоритм построения дизъюнктивных диаграмм по произвольной КНФ и основанный на DJD алгоритм препроцессинга КНФ при помощи SAT-оракула, описанный в п. 2.3.2 диссертации. Отметим, что в экспериментах раздела 3.4 использовался упрощенный вариант DJD-препроцессинга, в рамках которого для пути $\pi \in \Pi_?$, такого что $C(\pi)$ невыполнима (см. раздел 2.3.1), что доказывалось SAT оракулом, строилось новое ограничение (логическое следствие исходной КНФ), добавляемое к исходной КНФ, но перенаправление π в диаграмме в 1-вершину не осуществлялось. Данная модификация была сделана с целью уменьшения времени работы описанной процедуры препроцессинга. Отдельно подчеркнем, что порождаемые при помощи DJD дизъюнкты являются логическими следствиями исходной формулы и, таким образом, могут рассматриваться как примеры конфликтной информации.

Было проведено два эксперимента: в первом рассматривались трудные SAT-задачи с ежегодного соревнования SAT-решателей “SAT Competition 2021”, а во втором — экземпляры SAT, кодирующие задачи проверки эквивалентности булевых схем. В обоих экспериментах использовались 5 узлов вычислительного кластера (см. раздел 3.1.4 главы 3). В первом эксперименте в 14 из 26 тестов удалось достичь ускорения решения SAT-задач после использования препроцессинга. Стоит отметить, что, несмотря на близкий к 50% результат в контексте числа ускорившихся задач, суммарное время решения всех задач уменьшилось более чем в 2,4 раза. Детальное описание результатов эксперимента достаточно объемно и приведено целиком в тексте диссертации.

Во втором эксперименте в качестве тестовых задач использовались за-

[30]https://github.com/Vikseko/PyDJD.

Таблица 5 — Сравнение декомпозиционной трудности трудных LEC-задач до и после DJD-препроцессинга

Задача	**Декомпозиционная трудность**		
	до препроц. (с)	**после препроц. (с)**	**% ускорения**
CvK_{16}	2178663	1929530	11.4
CvW_{16}	1418200	1298746	8.4
DvC_{16}	1302856	1118082	14.1
DvK_{16}	2301015	2165651	5.8
DvW_{16}	1595147	—	0
KvW_{16}	2344223	2215808	5.4

дачи проверки эквивалентности алгоритмов умножения 16-битных чисел, описанные в п. 3.1.2 (CvK_{16}, CvW_{16}, DvC_{16}, DvK_{16}, DvW_{16}, KvW_{16}). Для каждой рассматриваемой формулы строилась оценка декомпозиционной трудности (методом, описанным в разделе 2.2.2 главы 2) до выполнения DJD-препроцессинга, и после его выполнения 12 разными способами: с построением дизъюнктивной диаграммы по прямому порядку, по частотному и по 10 различным случайным порядкам. В результате для каждой КНФ, кроме DvW_{16}, был найден порядок, препроцессинг по которому улучшает базовую оценку декомпозиционной трудности. Далее было выполнено полное решение декомпозиции для получения точной декомпозиционной трудности для каждой задачи до выполнения препроцессинга и после его выполнения с лучшим из найденных по оценкам порядков. Результаты эксперимента приведены в таблице 5. Как мы можем видеть, для 5 из 6 рассматриваемых пар алгоритмов умножения препроцессинг позволил улучшить декомпозиционную трудность соответствующей задачи проверки эквивалентности на 5–14%, что можно считать позитивным результатом, поскольку рассматриваемые задачи являются экстремально трудными для всех известных современных SAT-решателей.

В **заключении** сформулированы основные результаты научной работы.

ОСНОВНЫЕ ВЫВОДЫ И РЕЗУЛЬТАТЫ РАБОТЫ

Выполненная работа посвящена разработке новых методов обработки конфликтных ограничений, генерируемых SAT-решателями, основанными на алгоритме CDCL.

В рамках проведенного исследования были получены следующие основные результаты:

1. Разработаны алгоритмы и техники, отслеживающие и использующие повторно порождаемые конфликтные ограничения для ускорения решения SAT при помощи алгоритма CDCL;
2. Разработанные алгоритмы реализованы в виде параллельного модульного SAT-решателя RADAR и нескольких однопоточных решателей, которые продемонстрировали хорошие результаты на экземплярах SAT с ежегодных соревнований SAT-решателей;
3. Разработаны полиномиальный алгоритм перенаправления путей в дизъюнктивной диаграмме, сохраняющий структуру диаграммы, и метод препроцессинга произвольных КНФ на основе данного алгоритма;
4. Алгоритмы работы с дизъюнктивными диаграммами реализованы в виде многопоточного программного комплекса для работы с дизъюнктивными диаграммами PyDJD, использующего SAT-оракул PySAT.

ОСНОВНЫЕ ПУБЛИКАЦИИ ПО ТЕМЕ ДИССЕРТАЦИИ

Статьи в журналах из перечня ВАК:

1. Кондратьев, В.С. Дубликаты конфликтных ограничений в CDCL-выводе и их использование в задачах обращения некоторых криптографических функций / В.С. Кондратьев, А.А. Семёнов, О.С. Заикин // Вычислительные методы и программирование. — 2019. — Т. 20. — С. 54–66.

Статьи в изданиях, индексируемых в Web of Science и Scopus:

2. Speeding Up CDCL Inference with Duplicate Learnt Clauses / S. Kochemazov, O. Zaikin, A. Semenov, V. Kondratiev // Proc. of ECAI 2020: 24th European Conference on Artificial Intelligence. Santiago de Compostela, Spain, 29 August – 8 September 2020 / Ed. by G. De Giacomo, A. Catala, B. Dilkina et al. — IOS Press, 2020. — Vol. 325. — P. 339–346.

3. Kondratiev, V. Using Decision Diagrams of Special Kind for Compactification of Conflict Data Bases Generated by CDCL SAT Solvers / V. Kondratiev, I. Otpuschennikov, A. Semenov // Proc. of MIPRO 2020: 43rd International Convention on Information, Communication and Electronic Technology. Opatija, Croatia, 28 September – 02 October 2020 / Ed. by M. Koricic et al. — IEEE, 2020. — P. 1046–1051.

4. Kondratiev, V. Using Disjunctive Diagrams for Preprocessing of Conjunctive Normal Forms / V. Kondratiev // Proc. of MIPRO 2022: 45th Jubilee International Convention on Information, Communication and Electronic Technology. Opatija, Croatia, 23–27 May 2022 / Ed. by N. Vrcek et al. — IEEE, 2022. — P. 883–887.

5. Measuring the Effectiveness of SAT-Based Guess-and-Determine Attacks in Algebraic Cryptanalysis / A. Gladush, I. Gribanova, V. Kondratiev et al. // Proc.

of Parallel Computational Technologies: 16th International Conference, PCT 2022. Dubna, Russia, 29–31 March 2022 / Ed. by L. Sokolinsky, M. Zymbler. — Springer, 2022. — Vol. 1618. — P. 143–157.

6. Kondratiev, V. Using Parallel SAT Solving to Study Hard Combinatorial Problems Associated with Boolean Circuits / V. Kondratiev, S. Kochemazov, A. Semenov // Proc. of Parallel Computational Technologies: 17th International Conference, PCT 2023. St. Petersburg, Russia, 28–30 March 2023. — Springer, 2023. — **(Статья принята к печати)**.

7. Kondratiev, V. Speeding up the Solving of Logical Equivalence Checking Problems with Disjunctive Diagrams / V. Kondratiev, I. Otpuschennikov, A. Semenov // Proc. of MIPRO 2023: 46rd International Convention on Information, Communication and Electronic Technology. Opatija, Croatia, 22–26 May 2023. — IEEE, 2023. — **(Статья принята к печати)**.

Статьи в других изданиях:

8. Кондратьев, В.С. Использование повторно порождаемых конфликтных ограничений в CDCL выводе для ускорения обращения некоторых криптографических хеш-функций / В.С. Кондратьев // Информационные технологии: Материалы 56-й Междунар. науч. студ. конф. Новосибирск, 22–27 апреля 2018 г. — Новосибирск: ИПЦ НГУ, 2018. — С. 108.

9. Кондратьев, В.С. Применение дубликатов конфликтных ограничений для ускорения работы алгоритмов решения проблемы булевой выполнимости / В.С. Кондратьев // Винеровские чтения: Материалы Всерос. молодежной науч.-практ. конф. Иркутск, 06–07 июня 2019 г. — Иркутск: ИРНИТУ, 2019. — С. 19–25.