DRAFT 12 апреля 2022 г.

Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

РАЗРАБОТКА ПАРАЛЛЕЛЬНЫХ АЛГОРИТМОВ РЕШЕНИЯ ЗАДАЧИ БУЛЕВОЙ ВЫПОЛНИМОСТИ С ИСПОЛЬЗОВАНИЕМ ПОИСКА ВЕРОЯТНОСТНЫХ ЛАЗЕЕК

Автор: Джиблави Ибрагим Билалович _____

Направление подготовки: 01.03.02 Прикладная математика и информатика

Квалификация: Бакалавр

Руководитель ВКР: Чивилихин Д.С., к.т.н. _____

Санкт-Петербург, 2022 г.

Обучающийся Джиблави Ибрагим Билалович
Группа М3439 Факультет ИТиП

Направленность (профиль), специализация
Математические модели и алгоритмы в разработке программного обеспечения

ВКР принята «____» _____ 20___ г.

Оригинальность ВКР _____%

ВКР выполнена с оценкой

Дата защиты «15» июня 2022 г.

Секретарь ГЭК Павлова О.Н. _____

Листов хранения

Демонстрационных материалов/Чертежей хранения

Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

УТВЕРЖДАЮ

Руководитель ОП

проф., д.т.н. Парфенов В.Г. _____

«____» _____ 20___ г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

Обучающийся Джиблави Ибрагим Билалович

Группа М3439 **Факультет** ИТиП

Квалификация: Бакалавр

Направление подготовки: 01.03.02 Прикладная математика и информатика

Направленность (профиль) образовательной программы: Математические модели и алгоритмы в разработке программного обеспечения

Тема ВКР: Разработка параллельных алгоритмов решения задачи булевой выполнимости с использованием поиска вероятностных лазеек

Руководитель Чивилихин Д.С., к.т.н., ординарный доцент

2 Срок сдачи студентом законченной работы до: «15» мая 2022 г.

3 Техническое задание и исходные данные к работе

Ключевой задачей является разработка эффективного алгоритма решения задач булевой выполнимости. Алгоритм по заданной формуле должен возвращать информацию о её выполнимости. Подразумевается параллельный алгоритм, использующий поиск вероятностных лазеек для сведения задачи к набору более простых подзадач и последующему решению этих подзадач с применением существующих решателей.

4 Содержание выпускной квалификационной работы (перечень подлежащих разработке вопросов)

В данной работе разработаны алгоритмы поиска вероятностных лазеек на основе эволюционных алгоритмов. Разработаны различные подходы к решению на основе методов поиска вероятностных лазеек, исследована их эффективность. Проведено сравнение эффективности с существующими конкурентоспособными алгоритмами.

5 Перечень графического материала (с указанием обязательного материала)

Графические материалы и чертежи работой не предусмотрены

6 Исходные материалы и пособия

а) ресурсы из ИЗ.

7 Дата выдачи задания «31» января 2022 г.

Руководитель ВКР _____

Задание принял к исполнению _____ «31» января 2022 г.

Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

АННОТАЦИЯ
ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

Обучающийся: Джиблави Ибрагим Билалович

Наименование темы ВКР: Разработка параллельных алгоритмов решения задачи булевой выполнимости с использованием поиска вероятностных лазеек

Наименование организации, в которой выполнена ВКР: Университет ИТМО

ХАРАКТЕРИСТИКА ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

1 Цель исследования: цель

2 Задачи, решаемые в ВКР:

а) разработка эффективных алгоритмов поиска вероятностных лазеек.

б) разработка методов решения задачи булевой выполнимости с применением вероятностных лаззек.

в) исследование эффективности полученных алгоритмов.

г) Здесь возможно надо более подробный план или вообще просто про ”крутой решатель”?

3 Число источников, использованных при составлении обзора: 10

4 Полное число источников, использованных в работе: 19

5 В том числе источников по годам:

Отечественных			**Иностранных**		
Последние 5 лет	От 5 до 10 лет	Более 10 лет	Последние 5 лет	От 5 до 10 лет	Более 10 лет
0	0	0	5	5	9

6 Использование информационных ресурсов Internet: да, число ресурсов: 2

7 Использование современных пакетов компьютерных программ и технологий:

Пакеты компьютерных программ и технологий	**Раздел работы**
пакеты	где

8 Краткая характеристика полученных результатов

оценка результатов

9 Гранты, полученные при выполнении работы

вроде не было

10 Наличие публикаций и выступлений на конференциях по теме выпускной работы

вроде не было, если не считать paper

Обучающийся Джиблави И.Б. _____

Руководитель ВКР Чивилихин Д.С. _____

«____» _____ 20___ г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Задача булевой выполнимости является крайне важной NP-полной [6] алгоритмической задачей, так как к ней сводится большое число задач. Среди них:

— Проверка моделей, использующийся для формальной верификации параллельных систем с конечным числом состояний [1].
— Восстановление секретного ключа в алгоритме RSA [16].
— Разложение булевых матриц [2], часто применяемое в рекоммендательных системах.
— 0-1 задача о рюкзаке (широко известная задача класса NP).
— Многие задачи планирования, например, one shop scheduling [18], сводятся к SAT.
— Больше примеров?.

Для задачи булевой выполнимости, как и для любой NP-полной задачи пока не известно алгоритма, решающего её за полиномиальное от размера входа время. Однако, спектр применения этой задачи достаточно широк, чтобы возникала практическая польза от разработки эффективных решателей.

Алгоритмы решения задачи булевой выполнимости глобально делятся на две категории. Первая — последовательные решатели, то есть работающие в одном потоке исполнения. Существующие эффективные и широко распространенные подходы к построению таких решателей более подробно описаны в разделе 1.2. Вторая — параллельные решатели, нацеленные на максимальную утилизацию доступных ресурсов многоядерных и многопроцессорных машин для ускорения процесса решения. Более подробно они описаны в разделе 1.3.

Целью данной работы является построение эффективного параллельного алгоритма решения задачи булевой выполнимости на основе поиска вероятностных лазеек. В рамках исследований в paper данное семейство подходов показало положительные результаты, однако остается достаточно большое пространство как для построения более конкурентоспособной реализации, так и для расширения семейства подходов.

В рамках данной работы поставлены и выполнены следующие задачи:

— Разработка эффективного алгоритма поиска вероятностных лазеек на основе подходов, изложенных в paper.

— Адаптация набора существующих решателей для использования в рамках разрабатываемых алгоритмов.
— Разработка эффективного параллельного сервиса для решения задач с подстановками, допускающий обмен знаниями между решателями.
— Разработка схем решения с использованием вероятностных лазеек, как предложенных в paper, так и альтернативных.
— Исследование эффективности реализованных схем и сравнение с существующими паралелльными решателями.

В главе 1 содержатся необходимые понятия и определения, детально описаны поставленные в данной работе задачи, описан подробный план по их решению. Также рассмотрены существующие подходы к решению задач булевой выполнимости, как параллельные, имеющие непосредсвенное отношение к данной работе как конкуренты, так и последовательные, устройство которых необходимо понимать, так как на определенные свойства таких решателей делается упор в устранении недостатков предлагаемых алгоритмов.

В главе 2 детально изложены предлагаемые подходы к решению подзадач, возникших при разработке алгоритма. Описана схема поиска вероятностных лазеек, способ разделения задач булевой выполнимости и его применение в данной работе. Изложен подход к параллельному решению подзадач, указаны недостатки наивного подхода, предложено и обосновано решение данных недостатков. Далее описана структура програмного кода разработанного приложения.

В главе 3 описан процесс оценки качества и производительности полученного семейства алгоритмов. Предоставлены и разобраны результаты экспериментов как над частями алгоритма, так и над получившимся решатетем в целом. Сделаны выводы относительно конкурентоспособности рассматриваемых схем решения.

ГЛАВА 1. ОБЗОР

Содержание главы

1.1. Термины и понятия

Переработать определения? Лазейки точно плохие

Определение 1. **Булевой формулой** [1] называется формула логики высказываний, сорержащая логические переменные и пропозициональные связки $\wedge, \vee, \neg$. Множество логических переменных обозначается V.

Определение 2. **Булевой функцией** называется отображение $E\colon \mathcal{B}^n \to \mathcal{B}$, где $\mathcal{B} = \{0, 1\}$, а n – число различных переменных. Булева формула F задает булеву функцию E_F. Далее разница между этими понятиями для нас несущественна, поэтому всегда будет упоминаться булева функция E.

Определение 3. **Задача булевой выполнимости (SAT)** — проверить, существует ли $x \in \mathcal{B}^n$ такой, что выполняется $E(x) = 1$.

Теорема 4. (Кук, Левин) *Задача булевой выполнимости принадлежит классу NP-полных задач* [6].

Определение 5. **Решателем, алгоритмом A** называется алгоритм, принимающий на вход описание булевой формулы C и выдающий результат проверки булевой выполнимости соответствующей булевой функции — $A(C)$. Возможны следующие результаты:

— `0`, функция невыполнима.

— `1`, функция выполнима. В этом случае также может быть возвращено удовлетворяющее функции назначение переменных R: $E(V \mid R) = 1$.

— `?`, решатель не смог решить задачу либо вследствие его неполноты, либо из-за нехватки ресурсов, таких как время или память.

Обозначим за $S(A)$ множество булевых функций, разрешимых алгоритмом A за полиномиальное время.

Определение 6. **Означиванием, подстановкой** набора переменных $B \subseteq V$ называется отображение $\hat{b}\colon B \to \mathcal{B}$. Означивание можно применить к булевой функции E, результатом будет другая булева функция $E[B \mid \hat{b}]\colon \mathcal{B}^{|V|-|B|} \to \mathcal{B}$, возможно тождественная. Она получается из исходной путем частичной подстановки переменных, назначенных функцией $\hat{b}$. Множество всех означиваний множества переменных B обозначается $\hat{B}$.

[1] https://ru.wikipedia.org/wiki/Булева_формула

Определение 7. **Вывод последствий, Unit Propagation (UP)** — алгоритм, принимающий на вход формулу и назначение набора переменных. Возвращает спиков литералов (то есть значений переменых), выведенных из заданного назначения, или информацию о возникновении конфликта, то есть о невыполнимости формулы с заданным назначением. Эффективно реализован в Minisat [10].

Определение 8. **Лазейкой** называется множество $B \subseteq V$ такое, что для любого из $2^{|B|}$ означиваний $\hat{b}$ переменных из B выполняется $E[B \mid \hat{b}] \equiv 0$. Ясно, что если для функции E существует лазейка, то она невыполнима, то есть $E \equiv 0$.

Определение 9. **ρ-Лазейкой** называется множество $B \subseteq V$ такое, что выполняется

$$\left|\left\{ \hat{b} \mid \hat{b} \in \hat{B}, E[B \mid \hat{b}] \equiv 0 \right\}\right| \geqslant \rho \cdot 2^{|B|}.$$

Определение 10. (ε, δ) **аппроксимация лазейки, вероятностная лазейка**. Зафиксируем $\varepsilon, \delta \in (0, 1)$, пожмножество переменных $B \subseteq V$ и алгоритм A, и рассмотрим множество означиваний $\hat{B}$ в терминах ρ-лазейки.

— Введем на $\hat{B}$ равномерное распределение и зададим случайную величину

$$\xi_B(\hat{b}) = \left[E[B \mid \hat{b}] \in S(A) \right].$$

Ясно, что эта величина распределена по Бернулли с $p = \rho_B$.

— Тогда B является (ε, δ) аппроксимацией лазейки, если

$$\Pr\left[1 - \varepsilon \leqslant \frac{1}{N} \sum_{j=1}^{N} \xi_j \right] \geqslant 1 - \delta,$$

где $N \geqslant \frac{4 \ln 2/\delta}{\varepsilon^2}$. Данное условие согласно теореме Чернова paper обеспечивает тот факт, что аппроксимация ρ_B, вычисляемая по формуле

$$\hat{\rho}_B = \frac{1}{N} \sum_{j=1}^{N} \xi_j, \tag{1}$$

отклоняется от истинного значения ρ_B не более, чем на ε с вероятностью не менее $1 - \delta$.

1.2. Последовательные решатели

Понимание устройства распостраненных схем последовательных решателей играет крайне важную роль в данной работе, так как они прямо или косвенно используются в качестве решателей подзадач, возникающих в процессе работы описываемого алгоритма.

Опишем в общих чертах схему работы классических последовательных решателей. *Алгоритм Дэвиса-Патнема-Логемана-Лавленда (DPLL)* [7] [8] — полный алгоритм для решения задачи булевой выполнимости, основанный на поиске с возвратом. Данный алгоритм заключается в разбиении задачи на подзадачи путем последовательного присвоения переменным булевых значений и дальнейшей проверкой на отсутствие конфликтов. Часто в данном типе алгоритмов применяются дополнительные правила: *резолюция* — вывод значения переменных, значения которых однозначно определяют значение дизъюнкта, *исключение чистых переменных*, то есть переменных, входящих в формулу либо только с отрицанием, либо только без отрицания.

На основе алгоритма DPLL построен подход *CDCL (Conflict-Driven Clause Learning, управляемое конфликтами обучение дизъюнктам)* [19]. Данный подход используется во всех решателях, включенных в качестве компонентов в данной работе, так как является одним из самых распространенных и эффективных. Подход расширяет семейство DPLL несколькими техниками, среди которых анализ структуры конфликтов, запоминание конфликтных дизъюнктов, эвристики ветвления и другие. Тот факт, что данные решатели аккумулируют определенный вид знаний, крайне важен для портфельных решателей, описанных в разделе 1.3 и также использующихся в данной работе.

1.3. Параллельные решатели

Параллельные решатели нацелены на ускорение процесса решения засчет увеличения утилизации выделенных ресурсов, в частности и в особенности многоядерных и многопроцессорных систем. Рассмотрим некоторые подходы к реализации таких алгоритмов.

Портфельные решатели основаны на запуске диверсифицированного набора последовательных решателей, часто основанных на алгоритме CDCL, позволяющем осуществлять обмен знаниями [5]. Типичными примерами таких решателей являются hordesat [3] и painless [15]. Диверсификация решателей неизбежно приводит к выучиванию разных наборов дизъюнктов, уско-

ряя тем самым каждый из них по отдельности засчет обмена. В данной работе используется фреймворк painless, который предоставляет инструменты в том числе для создания портфельного решателя. В частности, портфельный решатель painless-mcomsps, основанный на последовательном решателе MapleCOMSPS [13], является победителем параллельного трека Sat Competition [2] и используется в предлагаемом алгоритме.

Может вообще убрать? Никак не используется *Алгоритмы локального поиска* [17]. Данный подход основан на параллельном изменении значений переменных, и в данной работе не используется.

Алгоритмы разделяй-и-властвуй [11]. В отличие от портфельных решателей, данное семейство алгоритмов не занимается параллельным решением одной и той же задачи. Напротив, исходная задача разбивается на подзадачи тем или иным способом, после чего полученные подзадачи решаются параллельно. Пример такого решателя основан на упомянутом фреймворке painless [14]. Основной проблемой данного подхода является тот факт, что подпространства поиска очень часто имеют значительно разную сложность решения. В данной работе также будет произведена попытка решить эту проблему: предлагаемый в данной работе алгоритм относится к данному семейству решателей и является совершенно новым в том смысле, что на момент разработки не существует полноценных конкурентоспособных алгоритмов, основанных на идее разделения задачи на подзадачи через поиск вероятностных лазеек. Тем не менее, результаты, продемонстрированные в работе paper, дают основания полагать, что на основе этой идеи можно создать решатель, имеющий высокую производительность.

Выводы по главе 1

Заключение

[2] https://satcompetition.github.io/2021/slides/ISC2021-fixed.pdf

ГЛАВА 2. ТЕОРЕТИЧЕСКОЕ ИССЛЕДОВАНИЕ И АРХИТЕКТУРА

Содержание главы

2.1. Поиск вероятностных лазеек

В данном разделе описаны алгоритмы поиска вероятностных лазеек. Для поиска вероятностных лазеек применяются эволюционные алгоритмы, а именно, $(1+1)$ и $(\mu+\lambda)$ алгоритмы [4], [12]. Для полного описания теоретической схемы поиска вероятностных лазеек необходимо определить особь, фитнес-функцию, а также операторы скрещивания, мутации и отбора.

Особью во всех реализованных схемах поиска вероятностных лазеек является битовая маска $\overline{B}$, соответствующая множеству переменных B, включенных в лазейку.

Фитнес функция. В фитнес-функции используется аппроксимация значения ρ, формально определенная уравнением 1, так как такой подход позволяет вычислять его с достаточно высокой точностью, а главное — быстро. В качестве алгоритма A используется алгоритм вывода последствий (определение 7), эффективно реализованный в рамках решателя Minisat [10]. Данный алгоритм с точки зрения решателя не является полным, но имеет полиномиальное время работы.

Используется фитнес функция, описанная в paper. Её преимущество заключается в том, что она помимо максимизации $\hat{\rho}$-значения лазейки минимизирует её размер. Максимизация $\hat{\rho}$-значения достигается первой частью функции:

$$G_C\left(\overline{B}\right) = (1-\hat{\rho}_B)\cdot 2^{\omega|V|},$$

где C — формула 1, V — множество переменных в формуле, $w \in (0,1]$ – константа. Минимизация размера лазейки обеспечивается второй частью функции:

$$f_{C,\min|B|} = \hat{\rho}_B \cdot 2^{|B|}.$$

Действительно, при относительно близких значениях ρ большое влияние на функцию будет оказывать размер лазейки B. Итоговая фитнес функция выглядит следующим образом:

$$f_C\left(\overline{B}\right) = \hat{\rho}_B \cdot 2^{|B|} + (1-\hat{\rho}_B)\cdot 2^{\omega|V|}. \tag{2}$$

Операторы В качестве основного оператора мутации был выбран зарекоммендовавший себя в paper оператор Doerr [9]. В процессе разработки также использовался равномерный оператор мутации, однако результаты, которые он показывал, стабильно хуже результатов Doerr на всех примерах, поэтому он был отброшен, однако доступен в конфигурации. Реализованы операторы одноточечного и двуточечного скрещивания. Генетический алгоритм $(\mu + \lambda)$ был реализован аналогично схеме, предложенной в paper. Сравнение получившихся алгоритмов проведено в разделе 3.2.

2.2. Разбиение пространства поиска

В данном разделе описан основной подход к разбиению задачи на подзадачи, а так же реализованные подходы к использованию вероятностных лазеек в качестве опоры для разбиения задачи.

Для начала опишем общий подход к разделению задачи на подзадачи. Пусть есть формура E, содержащая переменные V. Зафиксируем подмножество переменных B. Тогда исходную задачу можно разбить на подзадачи, соответствующие всем $2^{|B|}$ подстановкам переменных из B. Решив каждую из этих подзадач, несложно сделать вывод и для исходной задачи:

— Если хотя бы одна из подзадач оказалась выполнимой, то и исходная формула выполнима.

— Если все подзадачи оказались невыполнимыми, то и исходная формула невыполнима.

Именно таким способом в этой работе производится разбиение задач. Однако, в качестве множества B берется не случайное множество, а множество, так или иначе полученное через поиск вероятностных лазеек, так как это, как будет показано данее, позволяет без непосредственного решения отсечь сразу большую долю подзадач.

Опишем базовый подход, использующий одну лазейку. Пусть, как описано в разделе 2.1, найдена лазейка $B \subseteq V$. Тогда пространство поиска можно разделить на $2^{|B|}$ подпространств, разделив тем самым задачу на $2^{|B|}$ независимых подзадач. Более того, хорошая вероятностная лазейка имеет аппроксимированное (а очень часто и точное) значение ρ близкое к единице. Это позволяет сразу отсечь большую долю невыполнимых подзадач, используя только метод вывода последствий. Схема разделения пространства поиска содержится на рисунке 1.

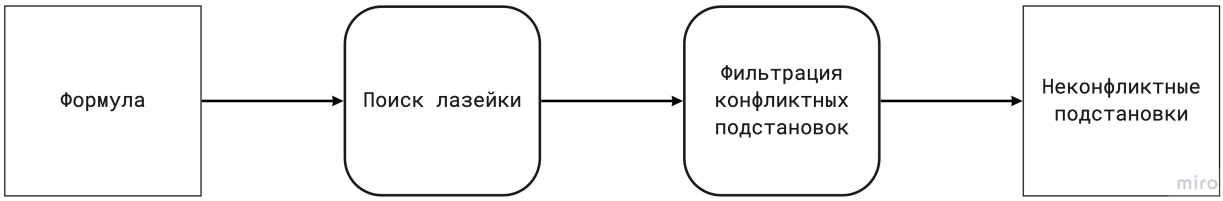

Рисунок 1 – Схема разбиения пространства поиска на основе одной лазейки

Более продвинутый подход для разбиения пространства поиска, описанный в paper, устроен следующим образом. Сначала находятся K различных лазеек $\{ B_i \subseteq V \}$. Далее для каждой лазейки независимо фильтруются подстановки, приводящие к конфликту. Множество неконфликтных подстановок, полученных для лазейки B_i, обозначим за C_i. Далее формируется декартово произведение $\times_i \{C_i\}$, представляющее собой предварительное разбиение пространства поиска. Далее это разбиение можно еще раз отфильтровать, избавившись от конфликтных подстановок. Смысл данного подхода заключается в том, что объединение неконфликтных подстановок (а именно из таких объединений состоит $\times_i \{C_i\}$) с немалой долей вероятности является конфликтной подстановкой, так как содержит назначение для большего числа переменных. В исследованиях, содержащихся в paper, данная догадка находит экспериментальное подтверждение, поэтому реализована в рамках данной работы. Данная схема проиллюстрирована на рисунке 2.

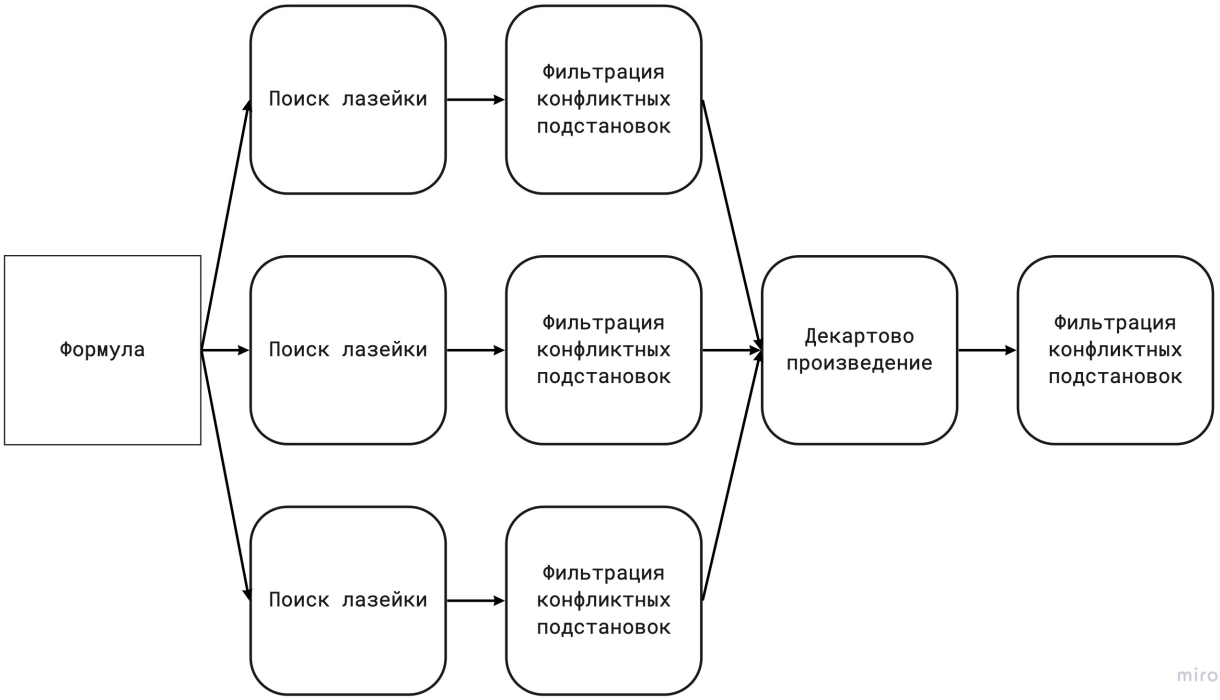

Рисунок 2 – Схема разбиения пространства поиска на основе декартова произведения

Стоит обратить внимание на тот факт, что данная схема является более масштабируемой, нежели предыдущая, так как искать различные лазейки можно параллельно, что и реализовано в данной работе. Сравнение производительности данных схем проведено в разделе 3.3.

2.3. Схемы решения

В данном разделе описаны способы решения задачи булевой выполнимости на основе разбиений пространства поиска, детально рассмотренных в разделе 2.2.

Прямое решение. В данной схеме, аналогичной подходу, используемому в paper, каждая подстановка переменных $\hat{b}$, полученная в результате разбиения задачи на подзадачи, решается каким-либо существующим решателем. В данной работе данный подход реализуется через сервис, описанный в разделе 2.5, что позволяет обеспечить параллельное решение с обменом знаниями между решателями.

Рекурсивное решение. Прямое решение имеет как минимум один критический для производительности и масштабируемости недостаток. Он заключается в том факте, что очень часто сложность подзадач из разбиения сильно ва-

рьируется. То есть, есть подзадачи, требующие большого количества ресурсов, главным образом, времени, а есть подзадачи, решающиеся очень быстро. Для трудных подзадач разумной идеей будет начать решение заново, то есть снова разбить подпространство поиска, соответствующее долго решаемой подстановке $\hat{b}$ на подпространства таким же образом, как это сделано для исходной задачи. Понятно, что такой подход можно продлить до любой глубины, однако осмысленно ограничить количество переразбиений подпространств.

Опишем теперь более детально схему предлагаемого подхода. Определим стек задач, состоящий из подстановок, подлежащих решению. Изначально этот стек состоит из одной пустой подстановки, соответствующей исходной задаче. Далее, пока стек не пуст или не найдено решение, снимаем со стека подстановку. Далее для данной подстановки, используя алгоритм, описанный в разделе 2.2, ищем разбиение на подзадачи. Для всех полученных подзадач запускаем решение, ограниченное по времени. Ограничение по времени может зависеть от глубины рекурсии, а так же может быть бесконечным, когда глубина рекурсии уже достаточно большая. Те подзадачи, которые не удалось решить за отведенное время, считаем трудными и кладем в стек для дальнейшего переразбиения и перерешивания.

Сравнение производительности подходов к решению подзадач описано в разделе 3.3.

2.4. Эффективный вывод последствий

Основным потребителем ресурсов в описанной схеме поиска вероятностных лазеек является алгоритм вывода последствий, реализованный в Minisat [10]. Поэтому оптимизация этого алгоритма и разработка новых подходов является крайне важной задачей для достижения высокой производительности.

Выборка. Заметим, что для достаточно маленьких B имеет смысл производить полный перебор множества $\hat{B}$ при вычислении $\hat{\rho}$. Во-первых, это обеспечит точное значение $\hat{\rho} = \rho$, во-вторых, реализация перебора всех подстановок (то есть, перебора всех возможных назначений переменных из набора в $\mathcal{B}$, что эквивалентно перебору всех чисел от 0 до $2^{|B|} - 1$ в двоичной записи) точно будет работать не медленнее, чем случайный перебор $2^{|B|}$ подстановок, так как не пользуется геренацией случайных чисел и делает строго меньше

операций засчет того, что не на каждом шаге модифицируются все значения подстановки.

Данное наблюдение создает необходимость в абстракции от типа выборки. Для этого был создан интерфейс Search, а также следующие реализации:

— FullSearch — перебор всех возможных $\hat{b} \in \hat{B}$.

— RandomSearch — перебор N случайных подстановок из $\hat{B}$.

— UniqueSearch — перебор N уникальных случайных подстановок из $\hat{B}$.

— CartesianSearch — перебор всех подстановок из декартового произведения нескольких наборов подстановок, необходимая для реализации некоторых стратегий (см. Главу 2.3).

Реализация FullSearch основана на алгоритме перебора двоичных чисел в диапазоне от 0 до $2^{|B|}$, а CartesianSearch на его обобщении на ‘числа’ с основанием, зависящим от положения знака.

На рисунке 3 представлена схема наследования интерфейса Search. Далее в таблице 1 описаны методы интерфейса.

Переделать картинки из Miro в inkscape? Чтобы без марки

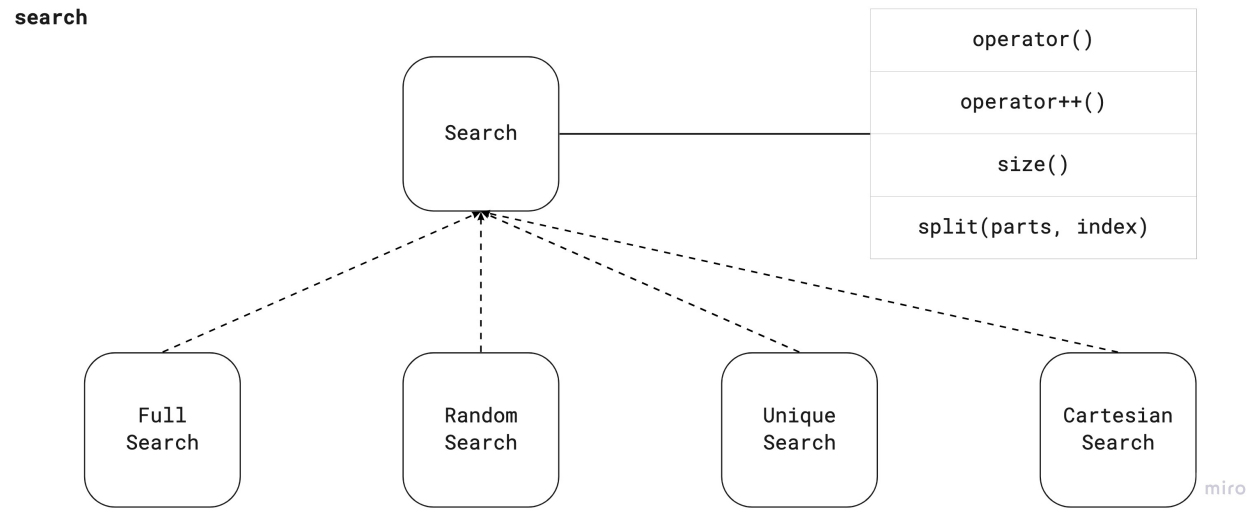

Рисунок 3 – Интерфейс Search и его реализации

Таблица 1 – Описание методов Search

Метод	Параметры	Описание
operator()	–	Возвращает текущий элемент выборки
operator++	–	Переходит на следующий элемент выборки. Возвращает false, если новый элемент – последний
size	–	Возвращает размер выборки
split	Число потоков, номер потока	Возвращает часть исходной выборки

После получения выборки необходимо вызвать метод вывода последствий на всех элементах этой выборки. Далее будут рассмотрены различные подходы к решению этой задачи: *наивный (последовательный), параллельный, вычисление ρ через дерево поиска*.

Наивный подход. Данный подход подразумевает последовательный вызов метода вывода последствий. То есть, имея выборку D и решатель A, значение ρ аппроксимируется так, как показано в листинге 1.

Листинг 1 – Наивное вычисление $\hat{\rho}$

```
function calculate_rho(A, D)
    S ← 0
    do
        b̂ ← D()
        if A(b̂) = 0 then
            S ← S + 1
        end if
    while D++
    return S/N
end function
```

Параллельная обработка выборки. Данный подход разделяет выборку на несколько выборок и обрабатывает их в разных потоках. Для этого используется метод split. Псевдокод данного подхода представлен в листинге 2.

Листинг 2 – Параллельное вычисление $\hat{\rho}$

function calculate_rho_par([A$_i$, $i = \overline{1,M}$], D)
    $S \leftarrow 0$
    [D$_i$] $\leftarrow$ [D.split(M, j), $j = \overline{1,M}$]
    return $\sum_i^{\text{parallel}}$ calculate_rho(A_i, D_i)$/N$
end function

Данный подход с теоретической точки зрения крайне хорошо масштабируется, так как метод split занимает несравнимо мало времени по сравнению с многократным вызовом метода вывода последствий. Практическое сравнение последовательного и параллельного подходов приведено в разделе 3.1.

Вычисление точного значения ρ через дерево подстановок. Данный подход является новым и не имеет аналогов в известных мне решателях за ненадобностью. Однако, идея, описанная далее, является крайне эффективной оптимизацией перебора $\hat{B}$, как показано в разделе 3.1. Для вычисления точного значения ρ необходимо посчитать число подстановок $\hat{b} \in \hat{B}$, которые решаются заданным алгоритмом A. В данной работе, как упомянуто ранее, в качестве этого алгоритма используется алгоритм вывода последствий. Результатом работы этого алгоритма может быть либо информация о существовании конфликтующих подстановок переменных, либо отсутствие дополнительной информации. Заметим, что если на каком-то префиксе $\hat{b}$ алгоритм сообщил о конфликте, то нет никакого смысла далее рассматривать подстановки с этим префиксом, так как все они заведомо конфликтные. Можно просто прибавить к результату размер всего поддерева поиска. Иллюстрация к данной идее приведена на рисунке 4. Листинг, содержащий псевдокод данного метода, приведен в приложении (листинг **??**). Отметим, что алгоритмы поиска вероятностных лазеек, описанные в разделе 2.1, чья эффективность напрямую зависит от эффективности вычисления ρ, ищут лазейки, значение ρ которых достаточно быстро становится близким к единице. Это наблюдение дает основания полагать, что производительность данного метода будет очень высокой, так как доминирующее число поддеревьев будет отсечено на ранней стадии. Данный метод был успешно реализован и встроен в исходный код решателя Minisat. Сравнительные результаты с наивным и параллельным подходами содержатся в разделе 3.1.

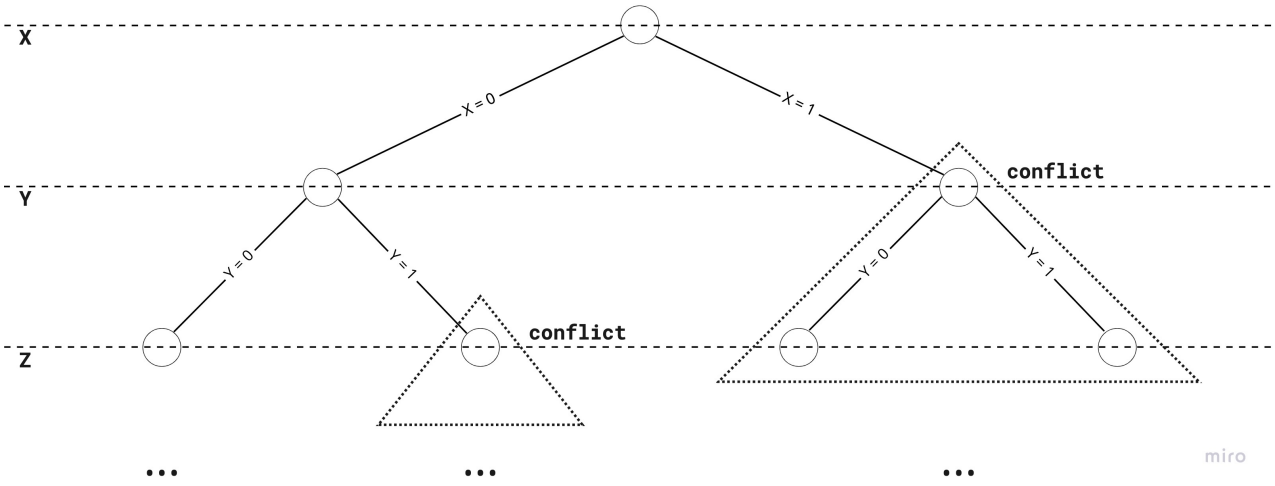

Рисунок 4 – Вычисление ρ через дерево подстановок. Здесь X,Y,Z — переменные. Подстановки $E[X \mid 1]$ и $E[\{ X,Y \} \mid \{ X \to 0, Y \to 1 \}]$ приводят к конфликту, поэтому соответствующие поддеревья поиска можно не рассматривать.

2.5. Решение набора подстановок

В данном разделе описана архитектура сервиса решения булевой функции с подстановками. Описан интерфейс сервиса, приведена схема распределения задач по последовательным решателям, описана техника обмена знаниями между решателями.

В качестве последовательных решателей используются Minisat [10], MapleCOMSPS [13], painless-maplecomsps [15]. Под последовательным имеется в виду не тип самого решателя, а невозможность параллельного решения разных задач. То есть, painless-mcomsps является параллельным решателем, но может решать лишь одну задачу одновременно. Описываемый сервис нужен именно для того, чтобы обеспечить возможность параллельного решения разных задач, что необходимо для реализации параллельных алгоритмов решения на основе подхода разделяй-и-властвуй.

Схема наследования интерфейса SolverService проиллюстрирована на рисунке 5. Описание методов интерфейса приведено в таблице 2.

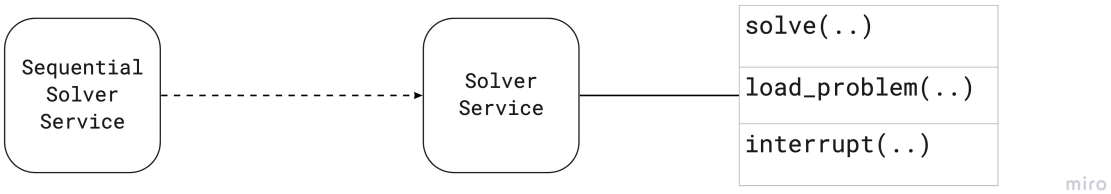

Рисунок 5 – Интерфейс SolverService и его реализация

Таблица 2 – Описание методов SolverService

Метод	Параметры	Описание
solve	Подстановка, ограничение по времени, callback-функция, вызываемая при окончании решения	Возвращает объект std::future типизированный результатом решения, и добавляет задачу в очередь
load_problem	–	Загружает формулу в решатель
interrupt	–	Прерывает процесс решения

Реализация SequentialSolverService управляет набором последовательных решателей, работающих параллельно: обеспечивает их задачами, а также, при необходимости, обеспечивает обмен знаниями. Для распределения задач реализована безопасная для использования в многопоточной среде очередь. Для обмена знаниями используются механизмы, реализованные в фреймворке painless [5] и адаптированные для использования разных решателей и схем обмена. Схема работы SequentialSolverService приведена на рисунке 6.

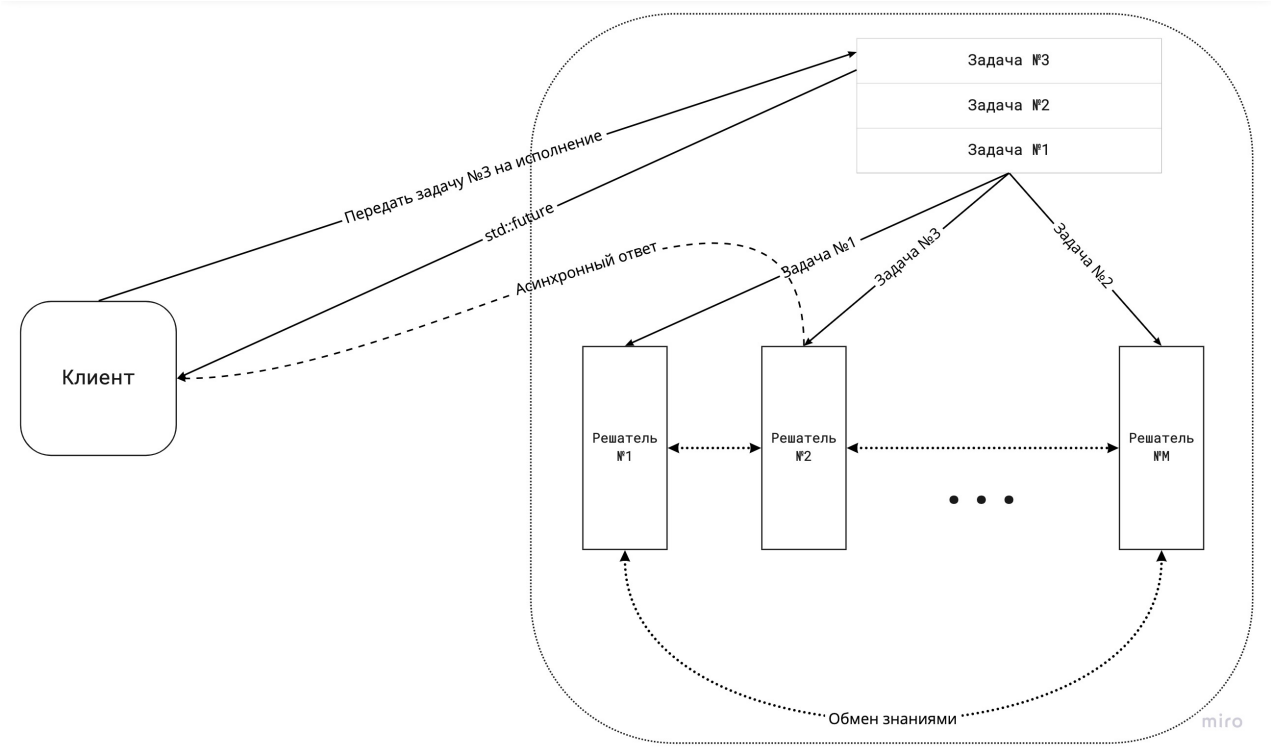

Рисунок 6 – Схема работы SequentialSolverService

Как уже было упомянуто, для обмена знаниями используются алгоритмы в составе фреймворка painless [15]. Также, решатель painless-mcomsps используется в качестве одного из вариантов реализаций последовательных решателей. В связи с этим стоит отметить, что данный фреймворк по своей надежности и качеству кода не удовлетворяет требованиям, которые требуются для описываемого сервиса. А именно, в процессе исправления и рефакторинга кода фреймворка было устранено большое множество ошибок, как логических, так и програмных. Из критических проблем стоит отметить как минимум две: MapleCOMSPS [1], приводящую к некорректному поведению в ряде редких случаев, а также некорректную реализацию основного метода класса Reducer [2], приводящую к неустойчивости решателя к прерываниям.

[1] https://github.com/vvallade/painless-sat-competition-2021/blob/b34579ca555a2989b610c3ac5df00711a76f2505/painless/mapleCOMSPS/mapleCOMSPS/core/Solver.cc#L1275

[2] https://github.com/vvallade/painless-sat-competition-2021/blob/b34579ca555a2989b610c3ac5df00711a76f2505/painless/painless-src/solvers/Reducer.cpp#L149

2.6. Описание реализации

Надо ли это вообще? В данном разделе описана структура исходного кода программы, перечислены возможные опции как при сборке, так и при запуске приложения.

Выводы по главе 2

Заключение

ГЛАВА 3. ПРАКТИЧЕСКОЕ ИССЛЕДОВАНИЕ

В данной главе описаны практические исследования, возникшие в процессе разработки алгоритма. В разделе 3.1 изложены эксперименты касательно эффективности предлагаемых алгоритмов вывода последствий. В разделе 3.2 — эксперименты, касающиеся эволюционных алгоритмов, описанных в разделе 2.1. Также в разделе 3.3 проведены замеры как базового решателя painless-mcomsps, так и разработанного в данной работе. Сделаны выводы о сильных и слабых сторонах полученного решения.

3.1. Алгоритмы вывода последствий

3.2. Эволюционные алгоритмы

3.3. Оценка производительности решателя

Выводы по главе 3

Заключение

ЗАКЛЮЧЕНИЕ

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 An Analysis of SAT-Based Model Checking Techniques in an Industrial Environment / N. Amla [et al.] // Correct Hardware Design and Verification Methods / ed. by D. Borrione, W. Paul. — Berlin, Heidelberg : Springer Berlin Heidelberg, 2005. — P. 254–268. — ISBN 978-3-540-32030-2.

2 *Avellaneda F.*, *Villemaire R.* Boolean Matrix Factorization with SAT and MaxSAT. — 2021. — DOI: 10.48550/ARXIV.2106.10105. — URL: https://arxiv.org/abs/2106.10105.

3 *Balyo T.*, *Sanders P.*, *Sinz C.* HordeSat: A Massively Parallel Portfolio SAT Solver // Theory and Applications of Satisfiability Testing – SAT 2015 / ed. by M. Heule, S. Weaver. — Cham : Springer International Publishing, 2015. — P. 156–172. — ISBN 978-3-319-24318-4.

4 *Borisovsky P.*, *Eremeev A.* A Study on Performance of the (1+1)-Evolutionary Algorithm. — 2003. — Jan.

5 Community and LBD-Based Clause Sharing Policy for Parallel SAT Solving / V. Vallade [et al.] // SAT 2020 - 23rd International Conference on Theory and Applications of Satisfiability Testing. Vol. 12178. — Alghero / Virtual, Italy, 07/2020. — P. 11–27. — (Lecture Notes in Computer Science). — DOI: 10.1007/978-3-030-51825-7_2. — URL: https://hal.inria.fr/hal-02906505 ; Due to the coronavirus COVID-19 pandemic, the conference was held virtually.

6 *Cook S. A.* The complexity of theorem-proving procedures // IN STOC. — ACM, 1971. — P. 151–158.

7 *Davis M.*, *Logemann G.*, *Loveland D.* A Machine Program for Theorem-Proving // Commun. ACM. — New York, NY, USA, 1962. — July. — Vol. 5, no. 7. — P. 394–397. — ISSN 0001-0782. — DOI: 10.1145/368273.368557. — URL: https://doi.org/10.1145/368273.368557.

8 *Davis M.*, *Putnam H.* A Computing Procedure for Quantification Theory // J. ACM. — New York, NY, USA, 1960. — July. — Vol. 7, no. 3. — P. 201–215. — ISSN 0004-5411. — DOI: 10.1145/321033.321034. — URL: https://doi.org/10.1145/321033.321034.

9 *Doerr B.*, *Doerr C.* Optimal Parameter Choices Through Self-Adjustment: Applying the 1/5-th Rule in Discrete Settings [Электронный ресурс]. — 2015. — URL: http://arxiv.org/abs/1504.03212.

10 *Eén N.*, *Sörensson N.* An Extensible SAT-solver // Theory and Applications of Satisfiability Testing / ed. by E. Giunchiglia, A. Tacchella. — Berlin, Heidelberg : Springer Berlin Heidelberg, 2004. — P. 502–518. — ISBN 978-3-540-24605-3.

11 *Elffers J.*, *Nordström J.* Divide and Conquer: Towards Faster Pseudo-Boolean Solving //. — 07/2018. — P. 1291–1299. — DOI: 10.24963/ijcai.2018/180.

12 *Lingaraj H.* A Study on Genetic Algorithm and its Applications // International Journal of Computer Sciences and Engineering. — 2016. — Oct. — Vol. 4. — P. 139–143.

13 MapleCOMSPS, MapleCOMSPS_LRB, MapleCOMSPS_CHB [Электронный ресурс] / J. H. Liang [et al.]. — 2016. — URL: https://docs.google.com/a/gsd.uwaterloo.ca/viewer?a=v%5C&pid=sites%5C&srcid=Z3NkLnV3YXRlcmxvby5jYXxtYXBsZXNhdHxneDo2YWEzYjEzN2JmY2I1Y (visited on 04/07/2022).

14 Modular and Efficient Divide-and-Conquer SAT Solver on Top of the Painless Framework / L. Le Frioux [et al.] // TACAS 2019 - 25th International Conference on Tools and Algorithms for the Construction and Analysis of Systems. Vol. 11427 / ed. by T. Vojnar, L. Zhang. — Prague, Czech Republic, 04/2019. — P. 135–151. — (Lecture Notes in Computer Science). — DOI: 10.1007/978-3-030-17462-0_8. — URL: https://hal.archives-ouvertes.fr/hal-02093520.

15 PaInleSS: a Framework for Parallel SAT Solving. / L. Le Frioux [et al.] // The 20th International Conference on Theory and Applications of Satisfiability Testing. Vol. 10491. — Melbourne, Australia : Springer, 08/2017. — P. 233–250. — (Lecture Notes in Computer Science). — DOI: 10.1007/978-3-319-66263-3_15. — URL: https://hal.archives-ouvertes.fr/hal-01540785.

16 *Patsakis C.* RSA private key reconstruction from random bits using SAT solvers // IACR Cryptol. ePrint Arch. — 2013. — Vol. 2013. — P. 26.

17 *Roli A.* Criticality and Parallelism in Structured SAT Instances // Principles and Practice of Constraint Programming - CP 2002 / ed. by P. Van Hentenryck. — Berlin, Heidelberg : Springer Berlin Heidelberg, 2002. — P. 714–719. — ISBN 978-3-540-46135-7.

18 Solving Open Job-Shop Scheduling Problems by SAT Encoding / M. KOSHIMURA [et al.] // IEICE Transactions on Information and Systems. — 2010. — Vol. E93.D, no. 8. — P. 2316–2318. — DOI: 10.1587/transinf.E93.D.2316.

19 *Zhang L.*, *Malik S.* The Quest for Efficient Boolean Satisfiability Solvers // Computer Aided Verification / ed. by E. Brinksma, K. G. Larsen. — Berlin, Heidelberg : Springer Berlin Heidelberg, 2002. — P. 17–36. — ISBN 978-3-540-45657-5.