МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

МАТЕРИАЛЫ

XI-й Международной молодёжной научной конференции

«МАТЕМАТИЧЕСКОЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ИНФОРМАЦИОННЫХ, ТЕХНИЧЕСКИХ И ЭКОНОМИЧЕСКИХ СИСТЕМ»

Томск, 24–27 мая 2024 г.

Под общей редакцией
кандидата технических наук И.С. Шмырина

Томск
Издательство Томского государственного университета
2024

ББК 22.17– 22.19
УДК 519.2, 519.7, 519.8
Т78

Т78 Труды Томского государственного университета. – Т. 309. Серия физико-математическая: Математическое и программное обеспечение информационных, технических и экономических систем : материалы XI-й Международной молодѐжной научной конференции. Томск, 24–27 мая 2024 г. / под общ. ред. И.С. Шмырина. – Томск : Издательство Томского государственного университета, 2024. – 324 с.

ISBN 978-5-907890-15-2

Сборник содержит материалы XI-й Международной молодѐжной научной конференции «Математическое и программное обеспечение информационных, технических и экономических систем», проводившейся 24–27 мая 2024 г. на базе Института прикладной математики и компьютерных наук Томского государственного университета. Материалы сгруппированы в соответствии с работавшими на конференции секциями.

Для научных работников, преподавателей, аспирантов, магистрантов и студентов.

УДК 539.3.004
ББК 22.17-22.19

ISBN 978-5-907890-15-2

Заключение

В данной работе была представлена разработка и реализация приложения отслеживающего изменения в расписании. Основным требованием при реализации этого приложения являлось требование отправки уведомлений студентам о любых изменениях расписания, таких как переносы и отмены занятий, смены аудиторий или преподавателей.

В процессе работы было разработано и реализовано мобильное приложение, способное показывать расписание на текущую неделю для студентов трех томских ВУЗов: ТГУ, ТПУ и ТУСУРа. Была добавлена возможность получать уведомления прямо на устройство пользователя о любых изменениях в расписании группы, указанной им при регистрации.

Несмотря на достигнутые результаты, данная работа оставляет место для дальнейшего совершенствования. Для этого планируется загрузить программу на сервер для беспрерывной работы, а также добавить возможность отслеживания расписания преподавателей и просмотра графика не только текущей недели, но и других. Помимо этого, хорошим дополнением к функциям приложения стала бы опция добавления в расписание личных дел и планов пользователя.

В заключение можно сказать, что разработанное приложение, отслеживающее изменения расписания и своевременно сообщающее об этом студентам, очень полезно для учебного процесса. Оно помогает тем, кто хочет вовремя получать актуальную информацию о смене графика, но не имеет возможности или желания постоянно отслеживать информацию на сайте. Такое приложение позволяет сэкономить время и делает образовательный процесс более комфортным.

С полным кодом приложения можно ознакомиться по ссылке: https://github.com/Nastyand/Schedule.git

ЛИТЕРАТУРА

1. *Андреева А.А., Пахомова Е.Г.* Создание Telegram-бота, отслеживающего изменения в расписании // Материалы X-й Международной молодежной научной конференции "Математическое и программное обеспечение информационных, технических и экономических систем", Томск, 26-29 мая 2023 г. – Труды Томского государственного университета. Т. 308: Серия физико-математическая. – Томск, 2023. – С. 95 – 101.
2. Доля рынка мобильных операционных систем по всему миру [Электронный ресурс]. – URL: https://gs.statcounter.com/os-market-share/mobile/worldwide (дата обращения: 21.05.2024)
3. Роль Android [Электронный ресурс]. – URL: https://www.android.com/intl/ru_ru/everyone/enabling-opportunity/ (дата обращения: 21.05.2024)
4. Java vs Kotlin – большой обзор [Электронный ресурс]. – URL: https://www.sravni.ru/kursy/info/java-vs-kotlin/ (дата обращения: 21.05.2024)
5. Firebase Realtime Database [Электронный ресурс]. – URL: https://firebase.google.com/docs/database (дата обращения: 21.05.2024)

ОЦЕНКА СКОРОСТИ ВСТРАИВАНИЯ ВОДЯНЫХ ЗНАКОВ ПРИ ПОКАДРОВОМ И ВНУТРИКАДРОВОМ РАСПАРАЛЛЕЛИВАНИИ

Анжин В.А.

Томский государственный университет

viktor.anjin@gmail.com

Введение

Развитие технологий производства и доставки мультимедиа материалов существенно увеличило объѐмы их производства и потребления. При этом всѐ более актуальной становится задача защиты авторских прав на цифровой контент, несанкционированное копирование и распространение которого приводит к потере доходов производителей или легальных дистрибьюторов.

Одним из подходов к защите авторских прав на мультимедиа материалы является использование водяных знаков. В случае утечки материала водяные знаки могут быть использованы для ограничения распространения пиратской копии. При внедрении уникальной для каждого из клиентов метки цифровые водяные знаки могут использоваться для поиска клиента, допустившего несанкционированное копирование и распространение материала [1].

Встраивание водяных знаков в изображение может осуществляться в разных вариантах представления данных, называемых областями внедрения [2]. Алгоритмы, использующие пространственную область, непосредственно манипулируют значениями яркости пикселей картинки. Алгоритмы, использующие частотную область, осуществляют внедрение водяного знака за счѐт корректировки частотных характеристик, и требуют предварительного перевода изображения в частотную область. В общем случае алгоритмы и пространственной, и частотной области внедрения требуют побайтовой обработки данных кадра, что делает встраивание водяных знаков ресурсоѐмкой задачей. Скорость встраивания водяного знака при работе на многопроцессорной системе может быть увеличена за счет организации параллельной обработки [3].

В рамках данной работы рассматриваются подходы к распараллеливанию реализаций алгоритмов встраивания водяных знаков. В первой части описывается алгоритм E_BLIND/D_LC [4], выбранный для оценивания подходов к распараллеливанию. Во второй части рассматриваются и оцениваются подходы к внутрикадровому распараллеливанию. В третьей части сравнивается эффективность покадрового и внутрикадрового распараллеливания.

Исходный код на языке C++, реализованный в рамках исследования алгоритма встраивания водяного знака E_BLIND/D_LC, реализации различных подходов к его распараллеливанию и используемые при проведении тестов изображения доступны в [5].

1. Описание алгоритма E_BLIND/D_LC

Алгоритм E_BLIND/D_LC использует пространственную область для внедрения однобитного сообщения m в исходный кадр Co. Результатом работы является кадр Cw. Для встраивания сообщения используется шаблонный кадр Wr, имеющий размеры сторон, соответствующие исходному кадру Co. Значение Wm выбирается исходя из встраиваемого бита. Настройка свойств устойчивости и видимости [6] алгоритма осуществляется через выбор значения α, используемого для масштабирования значений шаблонного кадра Wr:

$$\mathrm{Wm} = \begin{cases} \mathrm{Wr}, & \mathrm{m} = 1, \\ -\mathrm{Wr}, & \mathrm{m} = 0, \end{cases} \quad \mathrm{Wa} = \alpha \cdot \mathrm{Wm}, \ \mathrm{Cw} = \mathrm{Co} + \mathrm{Wa}.$$

В рамках описанного процесса можно выделить подготовительный этап, в ходе которого выбирается шаблонный кадр, осуществляется его масштабирование и корректировка знака, и следующий за ним этап внедрения, заключающийся в попиксельном суммировании полученного на подготовительном этапе кадра Wm и исходного кадра Co. Набор действий, осуществляемых на этих этапах, отличается от одного водяного знака к другому, но этап внедрения в общем случае требует попиксельного обхода кадра и выполнения некоторых арифметических действий. Исходя из этого, можно сделать предположение, что скорость внедрения водяных знаков, использующих пространственную или частотную область при различных подходах к распараллеливанию, будет изменяться в соответствии с тенденциями изменения скорости внедрения, полученными при оценке реализации алгоритма E_BLIND/D_LC.

2. Оценка подходов внутрикадрового распараллеливания

Для осуществления эффективной параллельной реализации нужно разбить задачу на независимые подзадачи. В рамках алгоритма внедрения E_BLIND/D_LC воздействие, осуществляемое на некоторый пиксель, зависит только от него самого и соответствующего ему пикселя масштабированного шаблонного кадра Wa, и не оказывает влияния на изменения значений других пикселей. Это даѐт возможность произвольным образом разбивать кадр для осуществления его параллельной обработки. Рассмотрим разбиение кадра по группам горизонтальных строк и оценим его в сравнении с разбиением по группам столбцов (рис. 1).

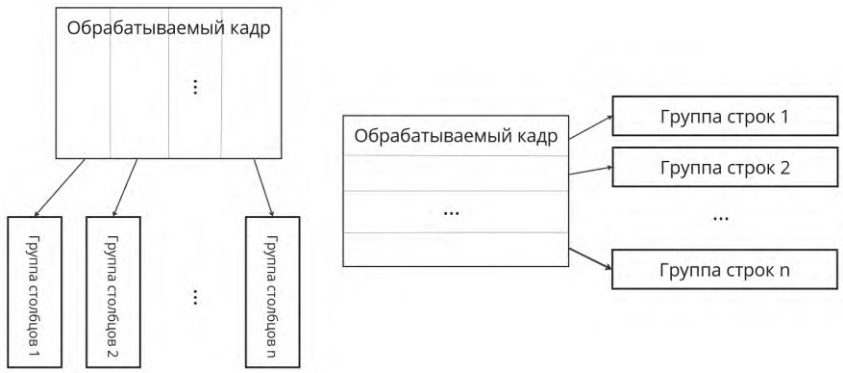

Рис 1. Разбиение кадра по столбцам(слева) и строкам(справа)

Для оценивания вариантов распараллеливания осуществлена программная реализация на языке C++, доступная по ссылке [5]. Измерения производительности проводились на компьютере с процессором Intel Core i7-12700, имеющем 12 независимых ядер и 20 логических процессоров. Оценка производилась на группах изображений различных размеров: группа SD включает изображения малого размера (размер варьируется около 640x480 пикселей), группа HD – изображения с размером, варьирующимся около 1920x1080 пикселей, и группа 4K – большие изображения размером более чем 4032x3000 пикселей. В столбцах табл. 1 указано количество параллельно работающих потоков, в ячейках – средние времена встраивания водяного знака в кадр в миллисекундах.

Таблица 1

Результаты замера скорости внедрения

	Количество потоков распараллеливания										
	1	**2**	**4**	**6**	**8**	**10**	**12**	**14**	**16**	**18**	**20**
SD столбцы	1.74	0.93	0.51	0.39	0.36	0.39	0.37	0.358	0.32	0.31	0.32
SD строки	1.75	0.93	0.49	0.37	0.35	0.39	0.35	0.35	0.34	0.38	0.33
HD столбцы	15.59	8.17	4.35	3.28	2.84	3.04	2.83	2.83	2.66	2.51	2.49
HD строки	15.62	8.18	4.32	3.21	2.75	2.96	2.75	2.77	2.60	2.46	2.41
4K столбцы	42.99	22.08	11.55	8.30	7.08	8.00	7.45	7.19	6.84	6.53	6.42
4K строки	43.08	22.32	11.48	8.29	6.98	7.79	7.36	7.10	6.75	6.44	6.31

Визуально результаты замера скорости, полученные в рамках эксперимента, представлены на рис. 2, отображающем зависимость скорости внедрения водяного знака исходя из типа распараллеливания (по группам строк и столбцов), размера кадра и количества потоков. Горизонтальная ось указывает число использующихся в параллельном режиме потоков. По вертикальной оси отложено время (в миллисекундах) внедрения водяного знака в кадр.

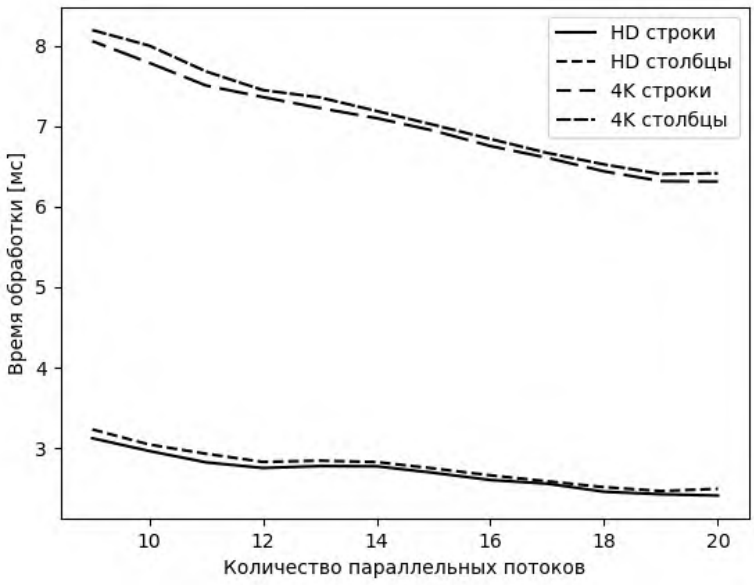

Рис 2. Зависимость скорости внедрения водяных знаков (в миллисекундах) от числа параллельных потоков

Из полученных результатов видно, что для изображений групп HD и 4K скорость внедрения при параллельной обработке по строкам выше скорости внедрения при параллельной обработке по столбцам. Одна из возможных причин полученных результатов заключается в том, что при параллельной обработке по строкам каждый из потоков имеет большую, по сравнению с параллельной обработкой по столбцам, область непрерывной памяти. За счѐт этого повышается вероятность попадания обрабатываемых данных в кэш-память, что приводит к ускорению работы [7]. В рамках проведѐнного эксперимента для маленьких картинок размером SD, тип внутрикадрового распараллеливания не оказывает влияния на скорость внедрения. Причина этого может заключаться в том, что накладные расходы распараллеливания и влияние сторонних приложений многопользовательской системы оказываются более существенными, чем различия в скорости между разными типами распараллеливания на маленьком размере картинок и используемом в эксперименте компьютере.

3. Оценка скорости при покадровом и внутрикадровом распараллеливании

Параллельная обработка может быть осуществлена как внутри кадра, так и на уровне отдельных кадров. Для организации внутрикадрового распараллеливания, как видно из проведѐнного ранее эксперимента, наиболее эффективно использовать разбиение на области последовательных строк. Параллельное внедрение на уровне отдельных кадров применимо для систем, имеющих в каждый момент времени несколько различных кадров для обработки.

В рамках оценки скорости внедрения осуществлена программная реализация алгоритма E_BLIND/D_LC с использованием покадрового и внутрикадрового распараллеливания на языке C++, доступная по ссылке [5]. Замеры производительности осуществлены на нескольких компьютерах и разных размерах кадров. Результаты в табл. 2 получены для процессора AMD Ryzen 7 5700U, имеющего 8 ядер и 16 независимых потоков и набора картинок размером 4K. В ячейках таблицы содержится среднее время (в миллисекундах) встраивания водяного знака в кадр. Строки таблицы указывают количество потоков покадрового распараллеливания, столбцы – внутрикадрового. Значение на пе-

ресечении строки n и столбца m указывает время, затраченное на внедрение водяного знака в кадр с использованием n потоков покадрового распараллеливания и m потоков внутрикадрового распараллеливания для каждого из этих кадров.

Таблица 2

Скорость внедрения при покадровом и внутрикадровом распараллеливании

		Количество потоков внутрикадрового распараллеливания								
		1	**2**	**4**	**6**	**8**	**10**	**12**	**14**	**16**
Количество потоков покадрового распараллеливания	**1**	13.432	7.161	7.157	5.364	4.580	3.923	3.492	3.238	3.081
	2	8.202	6.990	4.702	3.376	2.811	3.071	2.970	2.925	2.797
	4	5.617	4.261	2.780	2.793	2.767	2.757	2.725	2.717	2.731
	6	4.237	3.204	2.657	2.690	2.676	2.628	2.606	2.676	2.660
	8	3.776	2.757	2.753	2.738	2.695	2.693	2.665	2.656	2.672
	10	3.210	2.730	2.727	2.709	2.777	2.675	2.675	2.687	2.647
	12	3.011	2.700	2.646	2.573	2.570	2.522	2.570	2.530	2.496
	14	2.743	2.564	2.599	2.623	2.484	2.553	2.546	2.547	2.573
	16	2.569	2.453	2.439	2.404	2.390	2.512	2.439	2.391	2.481

Визуально результаты, полученные в рамках эксперимента, представлены ниже на рисунках, отображающих зависимость скорости внедрения от количества используемых потоков. По осям графиков отложено число потоков внутрикадрового и покадрового распараллеливания и время внедрения в миллисекундах.

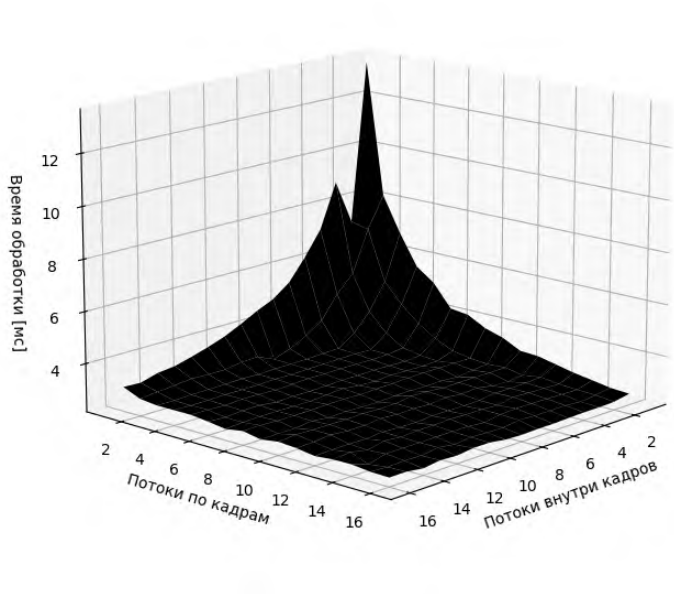

Рис 3. Зависимость скорости внедрения водяных знаков (в миллисекундах) от числа параллельных потоков (картинки группы 4K, AMD Ryzen 7 5700U)

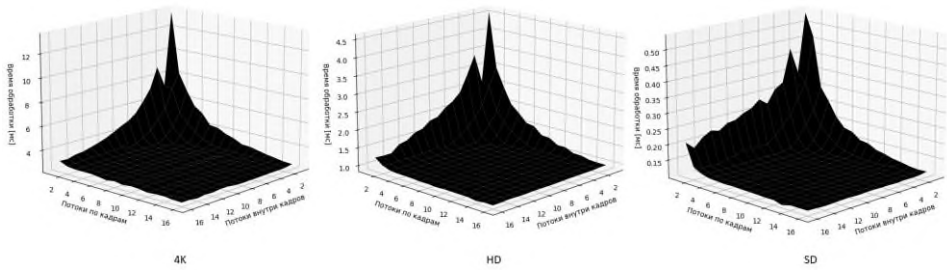

Рис 4. Зависимость скорости внедрения водяных знаков (в миллисекундах) от числа параллельных потоков (AMD Ryzen 7 5700U)

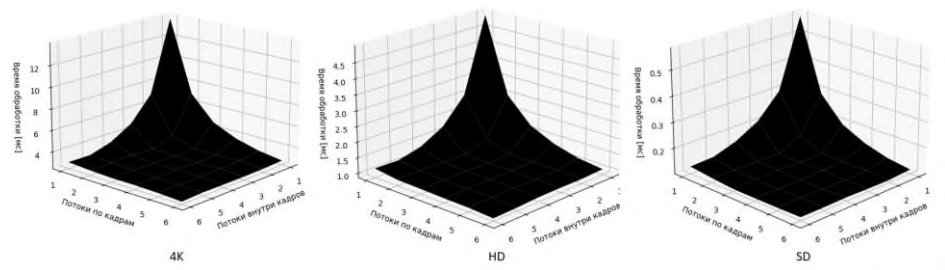

Рис 5. Зависимость скорости внедрения водяных знаков (в миллисекундах) от числа параллельных потоков (Intel Core i7-12700)

На рис. 6 отражена зависимость скорости внедрения водяного знака исходя из типа распараллеливания, размера кадра и количества потоков, полученная в результате эксперимента на компьютере с процессором AMD Ryzen 7 5700U. Горизонтальная ось указывает число использующихся в параллельном режиме потоков. По вертикальной оси отложено время (в миллисекундах) внедрения водяного знака в кадр. На рисунке изображена ситуация, в которой скорость внутрикадрового распараллеливания оказалась выше покадрового при параллельной обработке с использованием 2-х потоков. Подобная ситуация не была получена ни на каких других участвующих в эксперименте компьютерах.

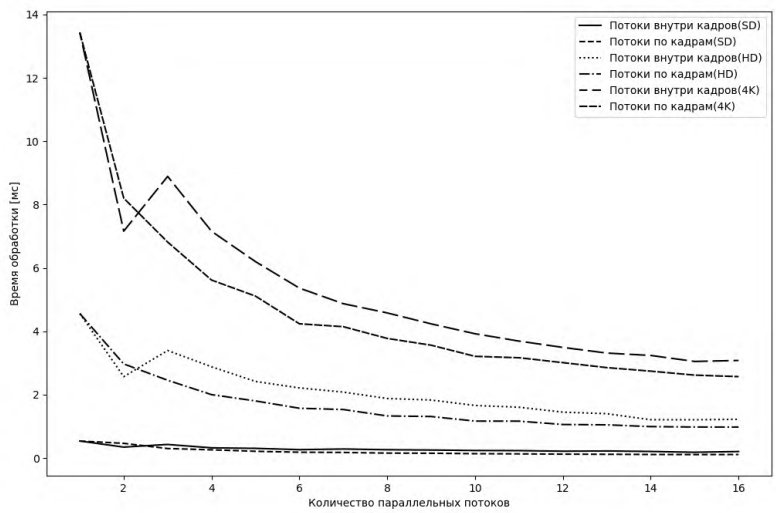

Рис 6. Зависимость скорости внедрения водяных знаков (в миллисекундах) от числа параллельных потоков

Из полученных в ходе эксперимента результатов видно, что скорость внедрения при параллельной обработке по кадрам выше скорости внедрения при параллельной внутрикадровой обработке. Но при малом количестве потоков, для ряда процессоров, скорость внутрикадрового распараллеливания может быть выше покадрового. Таким образом, если проектируемая система имеет возможность покадровой параллельной обработки, то целесообразно использовать её. Если количество возможных потоков покадрового распараллеливания меньше числа независимых потоков процессора, то имеет смысл осуществлять внутрикадровое распараллеливание с целью максимально задействовать вычислительные ресурсы процессора.

Заключение

В рамках эксперимента получены оценки скорости внедрения водяного знака E_BLIND/D_LC с использованием различных подходов к распараллеливанию. Полученные результаты позволяют сформулировать следующие рекомендации по параллельному встраиванию водяных знаков.

При параллельной обработке данных внутри фрейма наиболее эффективным оказывается его разбиение по группам горизонтальных строк пикселей. При параллельной обработке, организованной по отдельным фреймам, скорость внедрения оказывается быстрее по сравнению с параллельной обработкой, организованной по данным внутри каждого из фреймов. Таким образом, если проектируемая система имеет возможность покадровой параллельной обработки, то целесообразно использовать её. Если количество возможных потоков покадрового распараллеливания меньше числа независимых потоков процессора, то имеет смысл осуществлять внутрикадровое распараллеливание с целью максимально задействовать вычислительные ресурсы процессора.

ЛИТЕРАТУРА

1. *Wagner N.R.* Fingerprinting // 1983 IEEE Symposium on Security and Privacy. – IEEE, 1983. – P. 18.
2. *Орешкина Е.И., Фаворская М.Н.* Классификация методов нанесения цифровых водяных знаков // Актуальные проблемы авиации и космонавтики. – 2015. – Т. 1. – №. 11. – С. 414–415.
3. *Гергель В.П.* Теория и практика параллельных вычислений. – 2007.
4. *Cox I. et al.* Digital watermarking and steganography. – Morgan kaufmann, 2007.
5. Watermarks embedding performance evaluation and optimization // GitHub URL: https://github.com/anjin-viktor/watermarking_performance (дата обращения: 05.05.2024).
6. *Asikuzzaman M., Pickering M.R.* An overview of digital video watermarking // IEEE Transactions on Circuits and Systems for Video Technology. – 2017. – V. 28. – №. 9. – P. 2131–2153.
7. *Kowarschik M., Weiß C.* An overview of cache optimization techniques and cache-aware numerical algorithms // Algorithms for memory hierarchies: advanced lectures. – 2003. – P. 213–232.

МИНИМИЗАЦИЯ КОРНЯ ЛОГИЧЕСКОГО УРАВНЕНИЯ КНФ = 1

Бирюков М.О., Андреева В.В.
Томский государственный университет
iilll@duck.com, avv.21@mail.ru

Введение

Задача выполнимости булевых формул (SAT-задача) – фундаментальная задача в области информатики и теории вычислительной сложности. Она является первой, для которой удалось доказать ее принадлежность к классу NP-полных задач в начале 1970-х гг. (при формулировке относительно КНФ). Она имеет широкое применение в ряде областей, включая искусственный интеллект, верификацию программного обеспечения, криптографию, проектирование интегральных схем [1]. Активные исследования в её направлении ведутся по сей день, ежегодно проводится международное соревнование SAT-решателей (программных реализаций алгоритмов решения SAT-задачи) [2].

Большой класс существующих полных алгоритмов решения SAT-задачи основывается на методе ветвей и границ (поиске с возвратом), при котором поиск корня логиче-