Федеральное государственное автономное образовательное учреждение
высшего образования «Длинное название образовательного учреждения
«АББРЕВИАТУРА»

На правах рукописи

Азимов Рустам Шухратуллович

Алгоритмы выполнения навигационных запросов к графам с использованием операций линейной алгебры

Специальность XX.XX.XX —
«Технология обработки, хранения и переработки злаковых, бобовых культур, крупяных продуктов, плодоовощной продукции и виноградарства»

Диссертация на соискание учёной степени
кандидата физико-математических наук

Научный руководитель:
уч. степень, уч. звание
Фамилия Имя Отчество

Город — 2020

Оглавление

Стр.

Стр.

Введение

В современном мире становится всё больше данных, которые требуют анализа. При этом графы являются одной из самых распространённых структур данных. С их помощью большие объемы информации компактно и удобно представляются для анализа и обработки. Графы используются в программной инженерии и биоинформатике, в телекоммуникациях, социальных сетях и сетевом анализе и т.д. Также, в настоящее время активно развиваются графовые базы данных, используемые для хранения и анализа больших объёмов данных в виде графов. Следует упомянуть самые популярные графовые базы данных: RedisGraph, Neo4j, HyperGraphDB и OrientDB. При этом важной является задача поиска различных путей в графах. Чтобы описать свойства искомых путей, необходимо задать определенные ограничения на них. Данные ограничения формулируются в виде запроса к графу, а ответом на запрос является информация о существовании соответствующих путей, удовлетворяющих данным ограничениям. Говорят, что такие запросы вычеслены в relational семантике. Кроме того, в некоторых областях, в качестве доказательства существования таких путей необходимо предъявить все или хотя бы один из них, тогда говорят, что такие запросы вычислены в all-path и single-path семантиках соответственно.

Для описания ограничений на пути в помеченном графе естественно использовать формальные грамматики над некоторым алфавитом. Заданная грамматика ограничивает множество слов, полученных конкатенацией меток на рёбрах рассматриваемых путей. В настоящее время активно исследуются ограничения в виде контекстно-свободных (КС) грамматик, так как они позволяют описывать широкий класс запросов (КС-запросов), более широкий, чем, например, регулярные выражения. Однако большинство существующих алгоритмов вычисления КС-запросов имеют низкую производительность на больших графах, что затрудняет их применение на практике.

Одним из распространённых способов улучшения производительности алгоритмов на графах является их переформулирование в терминах линейной алгебры. Для тех алгоритмов, которые позволяют найти такую формулировку, имеется возможность применить разреженное представление матриц и параллельные вычисления, в частности, на GPU. Так, например, Лейуань Ван

(Leyuan Wang) сформулировал алгоритм подсчёта количества треугольников в графе на языке линейной алгебры и реализовал полученный алгоритм на GPU. Полученная реализация показала лучшую производительность по сравнению с реализациями на CPU. Кроме того, такого рода алгоритмы зачастую просты в реализации, так как позволяют использовать существующие библиотеки линейной алгебры (GraphBLAS::SuiteSparse, cuSPARSE, cuBLAS, m4ri, Scipy и др.). Таким образом, перспективным является направление формулирования алгоритмов вычисления КС-запросов к графам на языке линейной алгебры.

Однако возможность использования линейной алгебры в задачах поиска путей с КС-ограничениями в графах в настоящее время не исследована. Таким образом, актуальной задачей является разработка алгоритмов вычисления КС-запросов к графам, использующих различные операции линейной алгебры, и исследование их свойств.

Множество работ посвящены переформулированию классических алгоритмов на графах в терминах операций линейной алгебры. Например, Айдын Булук (Aydin Buluç), Упасана Шридхар (Upasana Sridhar), Питер Чжан (Peter Zhang) и Арифул Азад (Ariful Azad) в своих работах показывают как можно свести к линейной алгебре такие алгортимы, как поиск в ширину, алгоритм Дейкстры, алгоритм Беллмана-Форда и поиск наибольшего паросочетания в двудольном графе. Кроме того, существует стандарт GraphBLAS, который определяет базовые строительные блоки на языке линейной алгебры для алгоритмов на графах. GraphBLAS основан на том, что графы могут быть представлены в виде матрицы смежности или матрицы инцидентности. Также, ввиду того, что данные на практике разрежены, целесообразно использовать разреженный формат матриц. Однако, не каждый алгоритм на графах можно переформулировать на языке линейной алгебры. Так, например, до сих пор не смогли это сделать для алгоритма поиска в глубину. Также, в настоящее время, такая формулировка не найдена и для алгоритмов вычисления КС-запросов на графах.

Лесли Вэлиант (Leslie Valiant) провёл исследование, посвященное синтаксическому анализу КС-языков с использованием матричных операций. Однако предложенный им алгоритм позволяет проводить анализ только над строками, что эквивалентно анализу лишь специальных, линейно помеченных, графов. Кроме того, Филип Брэдфорд (Philip Bradford) исследовал задачу поиска кратчайших путей в графе с КС-ограничениями. Но предложенные им алгоритмы хоть и сформулированы на языке линейной алгебры, но предназначены только

для частного случая КС-грамматик и/или специализированных графов. Таким образом, данные алгоритмы не применимы к поиску путей с произвольными КС-ограничениями в произвольных графах.

Задаче поиска путей с КС-ограничениями в произвольных графах посвящены работы таких учёных, как Семён Григорьев, Джелле Хеллингс (Jelle Hellings), Сяованг Чжан (Xiaowang Zhang) и Мартин Мюзиканте (Martin Musicante). В данных исследованиях используются подходы, основанные на различных алгоритмах синтаксического анализа (LR, LL, GLL, CYK). Однако, данные алгоритмы не представленны в терминах линейной алгебры. Впервые вопрос о возможности нахождения матричного алгоритма вычисления КС-запросов к графам исследовал Михалис Яннакакис (Mihalis Yannakakis). Он указывал, что алгоритм Вэлианта может быть расширен для обработки графов без циклов, однако сомневался в возможности расширения алгоритма Вэлианта до произвольных графов.

Таким образом, на текущий момент не существует алгоритма вычисления произвольных КС-запросов к произвольным графам, выраженного на языке линейной алгебры. Поэтому необходимо дальнейшее исследование возможности разработки таких алгоритмов.

Целью данной работы является исследование применимости линейной алгебры в задаче вычисления КС-запросов к графам.

Целью данной работы является исследование применимости линейной алгебры в задаче вычисления КС-запросов к графам и получение реализаций данных алгоритмов с улучшенной производительностью.

Целью данной работы является исследование применимости линейной алгебры в задаче вычисления КС-запросов к графам и получение более высокопроизводительных реализаций для данных алгоритмов с использованием параллельных вычислительных систем.

Достижение поставленной цели обеспечивается решением следующих **задач**:

1. Разработать подход к вычислению КС-запросов к графам в терминах линейной алгебры.
2. Разработать семейство алгоритмов, использующих предложенный подход и вычисляющих КС-запросы с relational, single-path и all-path семантиками.

3. Разработать семейство алгоритмов вычисления КС-запросов к графам, использующих предложенный подход, вычисляющих КС-запросы с relational, single-path и all-path семантиками, и не требующих преобразований входной КС-грамматики.
4. Реализовать предложенные алгоритмы на CPU и GPU, провести их экспериментальное исследование на синтетических и реальных данных, сравнить их между собой и с существующими реализациями.

Научная новизна:

1. Алгоритмы вычисления КС-запросов к графам, которые используют подход, предложенный в диссертации, отличаются от алгоритмов, предложенных в работах Семёна Григорьева, Сяованга Чжана, Джелле Хеллингса, Филиппа Брэдфорда и Мартина Мюзиканте тем, что первые сформулированы в терминах линейной алгебры и вычисляют произвольные КС-запросы к произвольным графам. С практической точки зрения, предложенный подход, позволяет применять широкий класс оптимизаций для вычисления матричных операций и дает возможность автоматически распараллеливать вычисления за счёт существующих решений.
2. Впервые получены алгоритмы вычисления произвольных КС-запросов к произвольным графам, сформулированные в терминах линейной алгебры.
3. Алгоритмы, предложенный в диссертации, отличается от алгоритмов, предложенных в работах Семёна Григорьева, Сяованга Чжана, Джелле Хеллингса, Филиппа Брэдфорда и Мартина Мюзиканте возможностью работать с произвольными КС-грамматиками без необходимости их преобразования. Таким образом, удаётся избежать значительного увеличения размеров входной грамматики, от которого напрямую зависит временная сложность данных алгоритмов.
4. Экспериментальное исследование алгоритмов вычисления КС-запросов с различной семантикой к произвольным графам, использующих операции линейной алгебры проводится впервые и позволяет судить о применимости на практике разработанных алгоритмов.

Теоретическая и практическая значимость работы. Теоретическая значимость диссертационного исследования заключается в разработке подхода к вычислению КС-запросов к графам, использующего операции линейной ал-

гебры, в разработке формальных алгоритмов, использующих данный подход, а также в формальном доказательстве завершаемости, корректности и оценок временной сложности разработанных алгоритмов.

Полученные в ходе исследования реализации позволили оценить применимость на практике алгоритмов вычисления КС-запросов, сформулированных в терминах линейной алгебры. Кроме того, данные реализации могут быть использованы для интеграции с такими графовыми базами данных, как RedisGraph. Это добавит возможность вычислять КС-запросы к этим базам данных.

Методология и методы исследования. Методология исследования основана на линейной алгебре и теории графов. Подход, предлагающий использовать операции линейной алгебры при анализе графов, начал активно развиваться со второй половины 20-го века. В 2013 году был создан стандарта GraphBLAS, определяющий для алгоритмов на графах базовые строительные блоки на языке линейной алгебры. На текущий момент множество классических алгоритмов из теории графов были сформулированы в терминах линейной алгебры.

Кроме того, в исследовании использовалась теория формальных языков. Одной из задач, для которых до сих пор не найдена формулировка в терминах линейной алгебры, является поиск путей в графе с ограничениями в виде КС-грамматик. Данная задача использует подход к анализу строк, который начал активно развиваться в 50-х годах 20-го века в связи с изучением естественных языков (работы Н. Хомского). В последствии этот подход получил широкое распространение в различных областях, в том числе и связанных с анализом графов.

Доказательство завершаемости, корректности и оценок временной сложности предложенных алгоритмов проводится с применением линейной алгебры, теории формальных языков, теории графов и теории сложности алгоритмов.

Основные положения, выносимые на защиту:

1. Разработан подход к вычислению КС-запросов к графам в терминах линейной алгебры.
2. Разработано семейство алгоритмов, использующих предложенный подход и вычисляющих КС-запросы с relational, single-path и all-path семантиками. Доказана завершаемость и корректность предложенных

алгоритмов. Получена теоритическая оценка сверху временной сложности алгоритмов.

3. Разработано семейство алгоритмов, использующих предложенный подход и вычисляющих КС-запросы с relational, single-path и all-path семантиками, а также не требующих преобразований входной КС-грамматики. Доказана завершаемость и корректность предложенных алгоритмов. Получена теоритическая оценка сверху временной сложности алгоритмов.
4. Проведено экспериментальное исследование разработанных алгоритмов. Предложенные алгоритмы реализованы на CPU и GPU. Дальше детали.

Степень достоверности и апробация результатов. Достоверность и обоснованность результатов исследования опирается на использование формальных методов исследуемой области, выполнение формальных доказательств и инженерные эксперименты.

Основные результаты работы были представлены на ряде международных научных конференций: GRADES'18, GRADES'20, ADBIS'20, SIGMOD'21(еще не приняли). А также разработка предложенных алгоритмов была поддержана грантом РНФ №18-11-00100 и грантом РФФИ №19-37-90101.

Публикации. Все результаты диссертации изложены в 7 [**1**; **2**; **3**; **4**; **5**; **6**; **7**] научных работах. Из них 2 работы [**6**; **7**] индексируются ВАК и 6 работ [**1**; **2**; **3**; **4**; **5**; **6**] индексируются Scopus. Работы [**1**; **2**; **3**; **4**; **6**; **7**] написаны в соавторстве. В [**1**; **6**; **7**] Р. Азимову принадлежит разработка алгоритма, доказательство его корректности и завершаемости, реализация алгоритма, работа над текстом. В [**2**] Р. Азимову принадлежит разработка алгоритма, доказательство его корректности и завершаемости, работа над текстом. В [**3**; **4**] Р. Азимову принадлежит работа над доказательствами корректности и завершаемости алгоритма, работа над текстом.

Список литературы

[1] Azimov R. Context-Free Path Querying by Matrix Multiplication / Azimov R., Grigorev S. // In Proceedings of the 1st Joint International Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA) (GRADES-NDA'18)

[2] Azimov R. Context-Free Path Querying with Single-Path Semantics by Matrix Multiplication / Terekhov A., Khoroshev A., Azimov R., Grigorev S. // In Proceedings of the 3rd Joint International Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA) (GRADES-NDA'20)

[3] Azimov R. Context-Free Path Querying by Kronecker Product / Orachev E., Epelbaum I., Azimov R., Grigorev S. // In Proceedings of the 24th European Conference on Advances in Databases and Information Systems (ADBIS'20)

[4] Azimov R. Context-Free Path Querying by Kronecker Product большая версия / Orachev E., Epelbaum I., Azimov R., Grigorev S. // In Proceedings of the (SIGMOD'21)

[5] Azimov R. Ненаписанная работа матричный алгоритм по всем путям

[6] Azimov R. Path Querying with Conjunctive Grammars by Matrix Multiplication / Azimov R., Grigorev S. //Programming and Computer Software. – 2019. – Vol. 45. – №. 7. – pp. 357-364.

[7] Азимов Р. Ш. Синтаксический анализ графов с использованием конъюнктивных грамматик / Азимов Р., Григорьев С. // Труды ИСП РАН, 2018, том 1 вып. 2, с. 3-4.

Объем и структура работы. Диссертация состоит из введения, 6 глав, заключения и 3 приложений. Полный объём диссертации составляет 53 страницы, включая 2 рисунка и 4 таблицы. Список литературы содержит 0 наименований.

Глава 1. Обзор

В данной главе введены основные термины и определения, используемые в работе, а также рассмотрены основные алгоритмы вычисления КС-запросов к графам и используемые в ходе исследования инструменты.

1.1 Линейная алгебра

В данном разделе вводится ряд обозначений, а также представляется основная информация из линейной алгебры.

Операции над матрицами

Операции над векторами

Определение полукольца. Матричное умножение на полукольце.

Произведение Кронекера.

1.2 Теория графов

В данном разделе вводится ряд обозначений, а также представляется основная информация из теории графов.

Определение 1.2.1. *Граф* $\mathcal{G} = \langle V, E, L \rangle$, где V — конечное множество вершин, E — конечное множество рёбер, т.ч. $E \subseteq V \times L \times V$, L — конечное множество меток на рёбрах.

В дальнейшем речь будет идти о конечных ориентированных помеченных графах. Мы будем использовать термин *граф* подразумевая именно конечный ориентированный помеченный граф, если только не оговорено противное.

Также мы будем считать, что все вершины занумерованы подряд с нуля. То есть можно считать, что V — это отрезок $[0, |V| - 1]$ неотрицательных целых чисел, где $|V|$ — размер множества V.

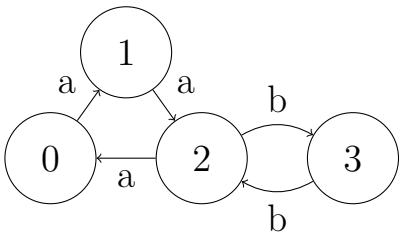


Рисунок 1.1 — Графическое представление графа $\mathcal{G}_1$.

Пример 1.2.1 (Пример графа и его графического представления)**.** Пусть дан граф

$$\mathcal{G}_1 = \langle \{0,1,2,3\}, \{(0,a,1), (1,a,2), (2,a,0), (2,b,3), (3,b,2)\}, \{a,b\} \rangle.$$

Графическое представление графа $\mathcal{G}_1$ изображено на рис. 1.1.

Определение 1.2.2. *Ребро* ориентированного помеченного графа $\mathcal{G} = \langle V, E, L \rangle$ это упорядоченная тройка $e = (v_i,l,v_j) \in V \times L \times V$.

Пример 1.2.2 (Пример рёбер графа)**.** $(0,a,1)$ и $(3,b,2)$ — это рёбра графа $\mathcal{G}_1$. При этом, $(3,b,2)$ $(2,b,3)$ — это разные рёбра, что видно из рис. 1.1.

Определение 1.2.3. *Путём* π в графе $\mathcal{G}$ будем называть последовательность рёбер такую, что для любых двух последовательных рёбер $e_1 = (u_1,l_1,v_1)$ и $e_2 = (u_2,l_2,v_2)$ в этой последовательности, конечная вершина первого ребра является начальной вершиной второго, то есть $v_1 = u_2$. Будем обозначать путь из вершины v_0 в вершину v_n как

$$v_0 \pi v_n = e_0,e_1,\ldots,e_{n-1} = (v_0,l_0,v_1),(v_1,l_1,v_2),\ldots,(v_{n-1},l_n,v_n).$$

Пример 1.2.3 (Пример путей графа)**.** $(0,a,1),(1,a,2) = 0\pi_1 2$ — путь из вершины 0 в вершину 2 в графе $\mathcal{G}_1$. При этом, $(0,a,1),(1,a,2),(2,b,3),(3,b,2) = 0\pi_2 2$ — это тоже путь из вершины 0 в вершину 2 в графе $\mathcal{G}_1$, но он не равен $0\pi_1 2$.

Кроме того, нам потребуется отношение, отражающее факт существования пути между двумя вершинами.

Определение 1.2.4. *Отношение достижимости* в графе: $(v_i,v_j) \in P \iff \exists v_i \pi v_j$.

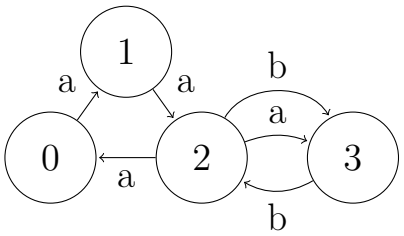
Отметим, что рефлексивность этого отношения часто зависит от контекста. В некоторых задачах по-умолчанию $(v_i,v_i) \notin P$, а чтобы это было верно, требуется явное наличие ребра-петли.

В данном исследовании мы будем задавать граф через его *матрицу смежности*.

Определение 1.2.5. *Матрица смежности* графа $\mathcal{G} = \langle V,E,L \rangle$ — это квадратная матрица M размера $n \times n$, где $|V| = n$ и ячейки которой содержат множества. При этом $l \in M[i,j] \iff \exists e = (i,l,j) \in E$.

Заметим, что наше определение матрицы смежности отличается от классического, в котором матрица отражает лишь факт наличия хотя бы одного ребра и, соответственно, является булевой. То есть $M[i,j] = 1 \iff \exists e = (i,\_,j) \in E$.

Пример 1.2.4 (Пример графа и его матрицы смежности)**.** Помеченный граф:



И его матрица смежности:

$$\begin{pmatrix} \varnothing & \{a\} & \varnothing & \varnothing \\ \varnothing & \varnothing & \{a\} & \varnothing \\ \{a\} & \varnothing & \varnothing & \{a,b\} \\ \varnothing & \varnothing & \{b\} & \varnothing \end{pmatrix}$$

Произведение графов

1.3 Теория формальных языков

В данном разделе вводится ряд обозначений, а также представляется основная информация из теории формальных языков.

Определение 1.3.1. *Алфавит* Σ — это конечное множество. Элементы этого множества будем называть *символами*.

Пример 1.3.1. Примеры алфавитов

- Латинский алфавит $\Sigma = \{a, b, c, \ldots, z\}$
- Кириллический алфавит $\Sigma = \{$а, б, в, ..., я$\}$
- Алфавит чисел в шестнадцатеричной записи

$$\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7,8,9, A, B, C, D, E, F\}$$

Традиционное обозначение для алфавита — Σ. Также мы будем использовать различные прописные буквы латинского алфавита. Для обозначения символов алфавита будем использовать строчные буквы латинского алфавита: $a, b, \ldots, x, y, z$.

Будем считать, что над алфавитом Σ всегда определена операция конкатенации $(\cdot) : \Sigma^* \times \Sigma^* \to \Sigma^*$. При записи выражений символ точки (обозначение операции конкатенации) часто будем опускать: $a \cdot b = ab$.

Определение 1.3.2. *Слово* над алфавитом Σ — это конечная конкатенация символов алфавита Σ: $\omega = a_0 \cdot a_1 \cdot \ldots \cdot a_m$, где ω — слово, а для любого i $a_i \in \Sigma$.

Определение 1.3.3. Пусть $\omega = a_0 \cdot a_1 \cdot \ldots \cdot a_m$ — слово над алфавитом Σ. Будем называть $m + 1$ *длиной слова* и обозначать как $|\omega|$.

Определение 1.3.4. *Язык* над алфавитом Σ — это множество слов над алфавитом Σ.

Пример 1.3.2. Примеры языков.

- Язык целых чисел в двоичной записи $\{0, 1, -1, 10, 11, -10, -11, \ldots\}$.
- Язык всех правильных скобочных последовательностей

$$\{(), (()), ()(), (())(), \ldots\}.$$

Любой язык над алфавитом Σ является подмножеством Σ^* — множества всех слов над алфавитом Σ. Заметим, что язык может является бесконечным множеством.

Определение 1.3.5. Грамматика G — это четвёрка $\langle \Sigma, N, P, S \rangle$, где

- Σ обозначает конечный алфавит терминальных символов или терминалов,
- N — алфавит нетерминальных символов или нетерминалов, $\Sigma \cap N = \varnothing$,
- P — конечное подмножество множества $(\Sigma \cup N)^+ \times (\Sigma \cup N)^*$,

– S — стартовый символ грамматики, $S \in N$.

Элемент $(a, b) \in P$ называется правилом вывода и записывается так: $a \to b$. При этом a называется левой частью правила, а b — правой частью. Левая часть любого правила из P обязана содержать хотя бы один нетерминал.

Определение 1.3.6. Вывод цепочки ω в грамматике G.
Цепочка $\omega_2 \in (\Sigma \cup N)^*$ *непосредственно выводима* из цепочки $\omega_1 \in (\Sigma \cup N)^+$ в грамматике $G = \langle \Sigma, N, P, S \rangle$ (обозначается $\omega_1 \to_G \omega_2$), если $\omega_1 = x_1 \cdot y \cdot x_2, \omega_2 = x_1 \cdot z \cdot x_2$, где $x_1, x_2, y \in (\Sigma \cup N)^*, z \in (\Sigma \cup N)^+$ и правило вывода $y \to z$ содержится в P. Индекс G в обозначении $\to_G$ обычно опускают, если G понятна из контекста.

Цепочка $\omega_2 \in (\Sigma \cup N)^*$ *выводима* из цепочки $\omega_1 \in (\Sigma \cup N)^+$ в грамматике G (обозначается $\omega_1 \Rightarrow_G \omega_2$), если существуют цепочки $z_0, z_1, \cdots, z_n (n \geqslant 0)$ такие, что $\omega_1 = z_0 \to z_1 \to ... \to z_n = \omega_2$. При этом последовательность $z_0, z_1, ..., z_n$ называется выводом длины n.

Определение 1.3.7. Язык, *порождаемый* грамматикой
$G = \langle \Sigma, N, P, S \rangle$ — это множество $L(G) = \{\omega \in \Sigma^* \mid S \Rightarrow \omega\}$.

Определение 1.3.8. *Деревом вывода цепочки* $\omega \in \Sigma^*$ в грамматике $G = \langle \Sigma, N, P, S \rangle$ называется упорядоченное дерево со следующими свойствами.

– Корень помечен S.
– Если его внутренний узел помечен $A \in N$ и $X_1, \ldots, X_k \in \Sigma \cup N$ — перечисленные слева направо пометки всех сыновей этого узла, то правило $A \to X_1 \ldots X_k \in P$.
– Если его внутренний узел помечен $A \in N$ и ε — пометка единственного сына этого внутреннего узла, то правило $A \to \varepsilon \in P$.
– $\omega = a_1 \ldots a_m$, где $a_1, \ldots, a_m \in \Sigma \cup \{\varepsilon\}$ перечисленные слева направо пометки всех листьев этого дерева.

Одним из распространённых способов классификации грамматик является иерархия по Хомскому. Грамматики типа 0 1 2 3. КС-грамматика.

Нормальная форма хомского для КС-грамматик.

Конечные автоматы.

Рекурсивные автоматы.

1.4 Постановка задачи вычисления КС-запросов к графам

В данном разделе вводится формальная постановка задачи вычисления КС-запросов к графам.

Выделим три возможных семантики запросов и приведем формальные постановки задачи вычисления КС-запросов для них.

Постановка задачи с реляционной семантикой запросов.

Постановка задачи с семантикой запросов одного пути.

Постановка задачи с семантикой запросов всех путей.

Общий мотивационный пример.

1.5 Существующие алгоритмы вычисления КС-запросов к графам

В данном разделе рассмотрены основные алгоритмы вычисления КС-запросов к графам.

Hellings/китайцы. Контекстно-свободные отношения R_A, и то что мы будем использовать эту формулировку в работе. Для всех семантик, всех графов. Временная сложность.

Брэдфорд. Кратчайшие пути, языки Дика, определенный вид графа. Временная сложность.

GLL, GLR, комбинаторы Merkat, все семантики, все графы. Временная сложность.

Бразильцы. Все графы. Временная сложность.

1.6 Используемые инструменты

В данном разделе рассмотрены инструменты, используемые в ходе исследования.

1.6.1 Стандарт GraphBLAS и его реализации

В данном разделе вводится ряд обозначений, а также представляется основная информация из стандарта GraphBLAS.

Стандарт GraphBLAS. Графы представлены в виде матриц смежности. Используются практические результаты для вычисления операций над разреженными матрицами.

Операции над матрицами и соответсвующие им трансформации графов. Транспонирование, поэлементное сложение/умножение, обычное умножение.

Матричное умножение на полукольце. Ключевая идея GraphBLAS.

Таблица стандартных полуколец в GraphBLAS.

Предствление матриц в GraphBLAS. CSR, CSC.

Таблица классических алгоритмов на графах с классической и матричными сложностями?

SuiteSparse — реализация на языке Си. Разреженные матрицы, их формат, вычисление операций на цпу.

1.6.2 Графовая база данных RedisGraph

RedisGraph — графовая база данных. Разреженные матрицы, их формат. Выразительность языка запросов.

1.6.3 Набор данных CFPQ_data

Созданный датасет. Описать какие графы RDF и запросы к ним. Таблица.

1.7 Выводы

На основе проведённого обзора можно сделать следующие выводы, обосновывающие необходимость проведения исследования в области вычисления КС-запросов к графам.

- Проблема вычисления КС-запросов к графам актуальна в нескольких областях: графовые базы данных, биоинформатика, статический анализ программ.
- Формулирование алгоритмов на графах в терминах операций линейной алгебры перспективное направление для улучшения производительности при работе с большими графами.
- Не проводилось исследований о возможность формулировки алгоритмов вычисления КС-запросов к графам в терминах операций линейной алгебры.

Обзор также позволяет выявить следующие подходы, технологии и средства.

- Для построения алгоритма КС-запросов к графам с использованием операций линейной алгебры целесообразно придерживаться стандарта GraphBLAS.
- Представление матриц должно быть разреженным.
- Для вычисления на цпу можно использовать реализацию SuiteSparse.
- В качестве хранилища данных можно использовать RedisGraph.
- В качестве датасета для экспериментального исследования можно использовать CFPQ_data.

Глава 2. Подход к вычислению КС-запросов к графам с использованием операций линейной алгебры

Подход. Все рассматриваем на примере.

Соотношение шага вывода для грамматики и операции над матрицами. Все над матрицами с элементами — множествами нетерминалов.

Переход к булевым полукольцам.

Сначала имеем матрицы с информацией о ребрах в графе (матрицы смежности). Потом, используя матричное умножение и другие операции для конкатенации путей в соответствии с шагом вывода грамматики. В результате будем иметь матрицы с информацией об искомых путях. По ним можно получить ответ на исходный КС-запрос.

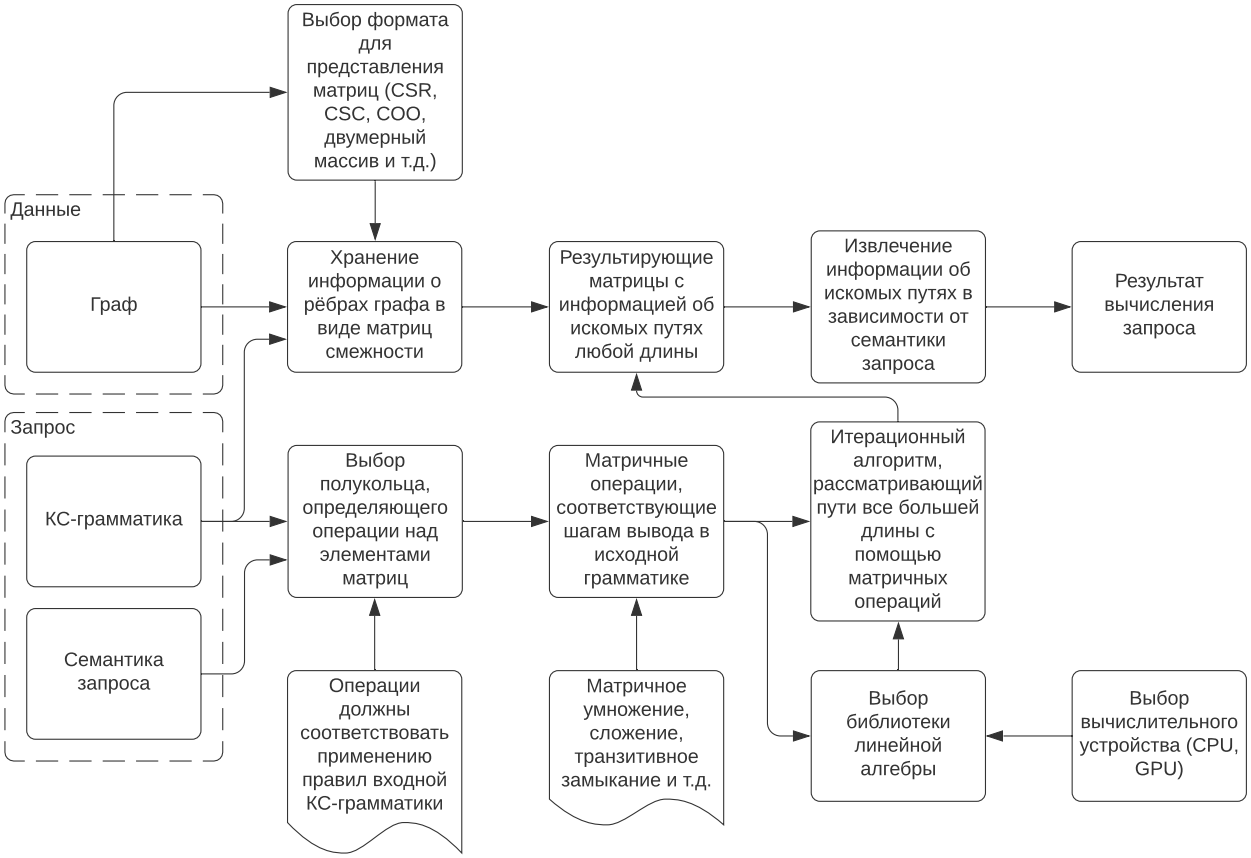


Рисунок 2.1 — Схема подхода к вычислению КС-запросов с помощью линейной алгебры.

Глава 3. Семейство алгоритмов вычисления КС-запросов с использованием матричного умножения

В данной главе изложены алгоритмы вычисления КС-запросов к графам с использованием матричного умножения и различных семантик запросов. Также сформулированы и доказаны утверждения о корректности изложенных алгоритмов. Кроме того, работа данных алгоритмов продемонстрирована на примере.

3.1 Алгоритмы

В данном разделе изложены алгоритмы вычисления КС-запросов к графам с использованием стандартных операций над булевыми матрицами и различных семантик запросов.

Определяем полукольцо.

Показываем переход к булевым матрицам. Обычное булевое матричное умножение и сложение.

Приводим базовый алгоритм для реляционной семантики запросов.

Вводим PathIndex. Опеределяем полукольцо.

Приводим алгоритм для одногу пути.

И алгоритм получения этого пути.

Приводим все тоже самое для семнтики всех путей.

3.2 Корректность алгоритмов

В данном разделе сформулированы и доказаны утверждения о корректности и завершаемости изложенных алгоритмов. Также представлены оценки временной сложности данных алгоритмов.

Теорема. Завершаемость всех алгортимов.

Теорема. Корректность базового алгоритма.

Теорема. Оценка временной сложности базового алгоритма.

Теорема. Корректность алгоритма одного пути.

Теорема. Корректность алгоритма получения пути.

Теорема. Оценка временной сложности алгоритма одного пути и алгоритма получения пути.

Теорема. Корректность алгоритма всех путей.

Теорема. Корректность алгоритма получения всех путей.

Теорема. Оценка временной сложности алгоритма всех путей и алгоритма получения путей.

3.3 Пример

В данном разделе работа изложенных алгоритмов продемонстрирована на примере, основанном на КС-языке правильных вложенных скобочных последовательностей.

Грамматика, граф.

Достигается худший случай.

Работа базового алгортима, картинки матриц. Результат.

Работа алгоритма одного пути. PathIndexes. Картинки матриц. Результат.

Работа алгоритма всех путей. Картинки матриц. Результат.

Глава 4. Семейство алгоритмов вычисления КС-запросов не требующих трансформации входной грамматики

В данной главе изложены алгоритмы вычисления КС-запросов к графам, который не требует трансформации входной КС-грамматики, а также использует операции линейной алгебры и различные семантики запросов. Также сформулированы и доказаны утверждения о корректности изложенных алгоритмов. Кроме того, работа данных алгоритмов продемонстрирована на примере.

Про необходимость создания алгоритма, не требующего преобр. грамматики. В других алгоритмах разрастается грамматика. Временная сложность зависит от размера грамматики.

4.1 Алгоритмы

В данном разделе изложены алгоритмы вычисления КС-запросов к графам, который не требует трансформации входной КС-грамматики, а также использует операции линейной алгебры и различные семантики запросов.

Строим по грамматике рекурсивный автомат.

Определяем полукольцо. Показываем переход к булевым матрицам. Произведение Кронекера.

Приводим алгоритм для реляционной семантики запросов.

Приводим алгоритм для одногу пути.

И алгоритм получения этого пути.

Приводим все тоже самое для семнтики всех путей.

4.2 Корректность алгоритмов

В данном разделе сформулированы и доказаны утверждения о корректности и завершаемости изложенных алгоритмов. Также представлены оценки временной сложности данных алгоритмов.

Теорема. Завершаемость всех алгортимов.

Теорема. Корректность алгоритма для реляционной семантики запросов.

Теорема. Оценка временной сложности алгоритма для реляционной семантики запросов.

Теорема. Корректность алгоритма одного пути.

Теорема. Корректность алгоритма получения пути.

Теорема. Оценка временной сложности алгоритма одного пути и алгоритма получения пути.

Теорема. Корректность алгоритма всех путей.

Теорема. Корректность алгоритма получения всех путей.

Теорема. Оценка временной сложности алгоритма всех путей и алгоритма получения путей.

4.3 Пример

В данном разделе работа изложенных алгоритмов продемонстрирована на примере, основанном на КС-языке правильных вложенных скобочных последовательностей.

Грамматика, граф.

Достигается худший случай.

Работа базового алгортима, картинки матриц. Результат.

Работа алгоритма одного пути. Картинки матриц. Результат.

Работа алгоритма всех путей. Картинки матриц. Результат.

Глава 5. Эксперименты, ограничения, обсуждение

Завершаемость и корректность предложенных алгоритмов формально доказаны выше, однако их производительность требует экспериментальной оценки. При этом основной интерес представляет оценка на входных данных, близких к реальным. Вместе с этим необходимо рассмотреть границы применимости полученных в работе результатов. Таким образом, в данной главе решаются следующие задачи.

Оценка производительности проводилась на реальных и на синтетических данных. Краткое описание экспериментов приведено в таблице.

Таблица эксперимент и что исследовалось. 1) Сравнение между матричными реализациями. 2) Сравнение лучших матричных с GLL и бразильцами.

Далее приводяся детальные описания экспериментов.

5.1 Экспериментальное исследование

5.1.1 Постановка экспериментов

Характеристики системы.

Описание реализаций.

Использованный датасет CFPQ_data.

5.1.2 Сравнение предложенных реализаций

Таблица, обсуждение.

5.1.3 Сравнение с существующими реализациями

Сравнение лучших матричных с GLL и бразильцами. Таблица. Обсуждение.

5.1.4 Выводы

Выводы

5.2 Ограничения

Используемые подходы и алгоритмы имеют ряд ограничений.

Пользуемся разреженностью реальных графов.

В случае когда грамматика нужна в нормальной форме, зависим от её разрастания.

Зависим от реализованных матричных операций. Функционала библиотек.

Большая временная сложность при определнном графе и грамматике.

Глава 6. Сравнение и соотнесение

В данной главе представлено сравнение полученных результатов с основными существующими решениями в области вычисления КС-запросов к графам. Описание существующих решений представлено в разделе 1.5 данной работы.

В качестве алгоритмов, с которыми производилось сравнение, выбраны следующие: GLL, Hellings, Bradford, бразильцы.

Для сравнения алгоритмов были выбраны критерии, представленные в таблице.

Заключение

Основные результаты работы заключаются в следующем.

1. Разработан подход к вычислению КС-запросов к графам, использующий операции линейной алгебры.
2. Разработано семейство алгоритмов вычисления КС-запросов к графам, использующих предложенный подход и позволяющих предоставлять искомые пути. Доказана завершаемость и корректность предложенных алгоритмов.
3. Разработано семейство алгоритмов вычисления КС-запросов к графам, использующий предложенный подход и не требующего преобразований входной КС-грамматики. Доказана завершаемость и корректность предложенных алгоритмов.
4. Проведено экспериментальное исследование разработанных алгоритмов.

Направления дальнейших исследований.

Благодарности.

Список сокращений и условных обозначений

a_n, b_n — коэффициенты разложения Ми в дальнем поле соответствующие электрическим и магнитным мультиполям

$\hat{\mathbf{e}}$ единичный вектор

E_0 амплитуда падающего поля

a_n, b_n — коэффициенты разложения Ми в дальнем поле соответствующие электрическим и магнитным мультиполям ещё раз, но без окружения minipage нет вертикального выравнивания по центру.

j тип функции Бесселя

k волновой вектор падающей волны

a_n, b_n — и снова коэффициенты разложения Ми в дальнем поле соответствующие электрическим и магнитным мультиполям, теперь окружение minipage есть и добавлено много текста, так что описание группы условных обозначений значительно превысило высоту этой группы... Для отбивки пришлось добавить дополнительные отступы.

L общее число слоёв

l номер слоя внутри стратифицированной сферы

λ длина волны электромагнитного излучения в вакууме

n порядок мультиполя

$\mathbf{N}_{e1n}^{(j)}$ $\mathbf{N}_{o1n}^{(j)}$ $\mathbf{M}_{o1n}^{(j)}$ $\mathbf{M}_{e1n}^{(j)}$ — сферические векторные гармоники

μ магнитная проницаемость в вакууме

r,θ,φ полярные координаты

ω частота падающей волны

BEM boundary element method, метод граничных элементов

CST MWS Computer Simulation Technology Microwave Studio программа для компьютерного моделирования уравнений Максвелла

DDA discrete dipole approximation, приближение дискретиных диполей

FDFD finite difference frequency domain, метод конечных разностей в частотной области

FDTD finite difference time domain, метод конечных разностей во временной области

FEM finite element method, метод конечных элементов

FIT finite integration technique, метод конечных интегралов

FMM fast multipole method, быстрый метод многополюсника

FVTD finite volume time-domain, метод конечных объёмов во временной области

MLFMA multilevel fast multipole algorithm, многоуровневый быстрый алгоритм многополюсника

MoM method of moments, метод моментов

MSTM multiple sphere T-Matrix, метод T-матриц для множества сфер

PSTD pseudospectral time domain method, псевдоспектральный метод во временной области

TLM transmission line matrix method, метод матриц линий передач

Словарь терминов

TeX : Система компьютерной вёрстки, разработанная американским профессором информатики Дональдом Кнутом

панграмма : Короткий текст, использующий все или почти все буквы алфавита

Список рисунков

Список таблиц

Приложение А

Примеры вставки листингов программного кода

Для крупных листингов есть два способа. Первый красивый, но в нём могут быть проблемы с поддержкой кириллицы (у вас может встречаться в комментариях и печатаемых сообщениях), он представлен на листинге А.1. Второй

Листинг А.1: Программа „Hello, world“ на C++

```
#include <iostream>
using namespace std;

int main() //кириллица в комментариях при xelatex и lualatex имеет проблемы с пробелами
{
  cout << "Hello, world" << endl; //latin letters in commentaries
  system("pause");
  return 0;
}

```

не такой красивый, но без ограничений (см. листинг А.2).

Листинг А.2: Программа „Hello, world“ без подсветки

```
#include <iostream>
using namespace std;

int main() //кириллица в комментариях
{
    cout << "Привет, мир" << endl;
}
```

Можно использовать первый для вставки небольших фрагментов внутри текста, а второй для вставки полного кода в приложении, если таковое имеется.

Если нужно вставить совсем короткий пример кода (одна или две строки), то выделение линейками и нумерация может смотреться чересчур громоздко.

В таких случаях можно использовать окружения `lstlisting` или `Verb` без `ListingEnv`. Приведём такой пример с указанием языка программирования, отличного от заданного по умолчанию:

```
fibs = 0 : 1 : zipWith (+) fibs (tail fibs)
```

Такое решение — со вставкой нумерованных листингов покрупнее и вставок без выделения для маленьких фрагментов — выбрано, например, в книге Эндрю Таненбаума и Тодда Остина по архитектуре

Наконец, для оформления идентификаторов внутри строк (функция `main` и тому подобное) используется `lstinline` или, самое простое, моноширинный текст (`\texttt`).

Пример A.3, иллюстрирующий подключение переопределённого языка. Может быть полезным, если подсветка кода работает криво. Без дополнительного окружения, с подписью и ссылкой, реализованной встроенным средством.

Листинг A.3: Пример листинга с подписью собственными средствами

```
## Caching the Inverse of a Matrix

## Matrix inversion is usually a costly computation and there
   may be some
## benefit to caching the inverse of a matrix rather than
   compute it repeatedly
## This is a pair of functions that cache the inverse of a
   matrix.

## makeCacheMatrix creates a special "matrix" object that can
   cache its inverse

makeCacheMatrix <- function(x = matrix()) {#кириллица в коммента
   риях при xelatex и lualatex имеет проблемы с пробелами
    i <- NULL
    set <- function(y) {
        x <<- y
        i <<- NULL
    }
    get <- function() x
    setSolved <- function(solve) i <<- solve
    getSolved <- function() i
    list(set = set, get = get,
    setSolved = setSolved,
    getSolved = getSolved)
```

```
}


## cacheSolve computes the inverse of the special "matrix" returned by
## makeCacheMatrix above. If the inverse has already been calculated (and the
## matrix has not changed), then the cachesolve should retrieve the inverse from
## the cache.

cacheSolve <- function(x, ...) {
    ## Return a matrix that is the inverse of 'x'
    i <- x$getSolved()
    if(!is.null(i)) {
        message("getting cached data")
        return(i)
    }
    data <- x$get()
    i <- solve(data, ...)
    x$setSolved(i)
    i
}
```

Листинг A.4 подгружается из внешнего файла. Приходится загружать без окружения дополнительного. Иначе по страницам не переносится.

Листинг A.4: Листинг из внешнего файла

```
# Analysis of data on Course Project at Getting and Cleaning data course of Data Science track at Coursera.

# Part 1. Merges the training and the test sets to create one data set.
# 3. Uses descriptive activity names to name the activities in the data set
# 4. Appropriately labels the data set with descriptive variable names.

if (!file.exists("UCI HAR Dataset")) {
    stop("You need 'UCI HAR Dataset' folder full of data")
}

```

```
library(plyr) # for mapvalues


#getting common data
features <- read.csv("UCI HAR Dataset/features.txt",sep=" ",
   header = FALSE,
                     colClasses = c("numeric","character"))
activity_labels <- read.csv("UCI HAR Dataset/activity_labels.txt
   ",sep="",
                            header = FALSE,colClasses = c("
   numeric","character"))

#getting train set data
subject_train <- read.csv("UCI HAR Dataset/train/subject_train.
   txt",
                          header = FALSE,colClasses = "numeric",
   col.names="Subject")
y_train <- read.csv("UCI HAR Dataset/train/y_train.txt", header
   = FALSE,
                    colClasses = "numeric")
x_train <- read.csv("UCI HAR Dataset/train/X_train.txt",sep="",
   header = FALSE,
                    colClasses = "numeric",col.names=features$V2
   ,check.names = FALSE)

activity_train <- as.data.frame(mapvalues(y_train$V1, from =
   activity_labels$V1,
                                            to = activity_labels$
   V2))
names(activity_train) <- "Activity"



#getting test set data
subject_test <- read.csv("UCI HAR Dataset/test/subject_test.txt"
   ,
                         header = FALSE,colClasses = "numeric",
   col.names="Subject")
y_test <- read.csv("UCI HAR Dataset/test/y_test.txt", header =
   FALSE,
                   colClasses = "numeric")
x_test <- read.csv("UCI HAR Dataset/test/X_test.txt",sep="",
   header = FALSE,
```

```
                    colClasses = "numeric",col.names=features$V2,
    check.names = FALSE)

activity_test <- as.data.frame(mapvalues(y_test$V1, from =
    activity_labels$V1,
                                         to = activity_labels$V2
    ))
names(activity_test) <- "Activity"


# Forming full dataframe
data_train <- cbind(x_train,subject_train,activity_train)
data_test <- cbind(x_test,subject_test,activity_test)
data <- rbind(data_train, data_test)

# Cleaning memory
rm(features, activity_labels, subject_train, y_train, x_train,
    activity_train,
    subject_test, y_test, x_test, activity_test, data_train, data
    _test)


# Part 2. Extracts only the measurements on the mean and
    standard deviation for each measurement.

cols2match <- grep("(mean|std)",names(data))

# Excluded gravityMean, tBodyAccMean, tBodyAccJerkMean,
    tBodyGyroMean,
# tBodyGyroJerkMean, as these represent derivations of angle
    data, as
# opposed to the original feature vector.

# Subsetting data frame, also moving last columns to be first
Subsetted_data_frame <- data[ ,c(562, 563, cols2match)]

# Part 5. From the data set in step 4, creates a second,
    independent tidy data set
# with the average of each variable for each activity and each
    subject.

library(dplyr) # for %>% and summarise_each
```

```
75 tidydata <- Subsetted_data_frame %>% group_by(Subject,Activity)
   %>%
            summarise_each(funs(mean))

write.table(tidydata, "tidydata.txt", row.names=FALSE)
```

Приложение Б

Очень длинное название второго приложения, в котором продемонстрировано работа с длинными таблицами

Б.1 Подраздел приложения

Вот размещается длинная таблица:

Параметр	Умолч.	Тип	Описание
&INP			
kick	1	int	0: инициализация без шума ($p_s = const$) 1: генерация белого шума 2: генерация белого шума симметрично относительно экватора
mars	0	int	1: инициализация модели для планеты Марс
kick	1	int	0: инициализация без шума ($p_s = const$) 1: генерация белого шума 2: генерация белого шума симметрично относительно экватора
mars	0	int	1: инициализация модели для планеты Марс
kick	1	int	0: инициализация без шума ($p_s = const$) 1: генерация белого шума 2: генерация белого шума симметрично относительно экватора
mars	0	int	1: инициализация модели для планеты Марс
kick	1	int	0: инициализация без шума ($p_s = const$) 1: генерация белого шума 2: генерация белого шума симметрично относительно экватора
mars	0	int	1: инициализация модели для планеты Марс
kick	1	int	0: инициализация без шума ($p_s = const$) 1: генерация белого шума 2: генерация белого шума симметрично относительно экватора
mars	0	int	1: инициализация модели для планеты Марс
kick	1	int	0: инициализация без шума ($p_s = const$) 1: генерация белого шума 2: генерация белого шума симметрично относительно экватора
mars	0	int	1: инициализация модели для планеты Марс
kick	1	int	0: инициализация без шума ($p_s = const$) 1: генерация белого шума 2: генерация белого шума симметрично относительно экватора
mars	0	int	1: инициализация модели для планеты Марс
kick	1	int	0: инициализация без шума ($p_s = const$) 1: генерация белого шума 2: генерация белого шума симметрично относительно экватора
mars	0	int	1: инициализация модели для планеты Марс
kick	1	int	0: инициализация без шума ($p_s = const$)

продолжение следует

(продолжение)			
Параметр	Умолч.	Тип	Описание
			1: генерация белого шума 2: генерация белого шума симметрично относительно экватора
mars	0	int	1: инициализация модели для планеты Марс
kick	1	int	0: инициализация без шума ($p_s = const$) 1: генерация белого шума 2: генерация белого шума симметрично относительно экватора
mars	0	int	1: инициализация модели для планеты Марс
kick	1	int	0: инициализация без шума ($p_s = const$) 1: генерация белого шума 2: генерация белого шума симметрично относительно экватора
mars	0	int	1: инициализация модели для планеты Марс
kick	1	int	0: инициализация без шума ($p_s = const$) 1: генерация белого шума 2: генерация белого шума симметрично относительно экватора
mars	0	int	1: инициализация модели для планеты Марс
kick	1	int	0: инициализация без шума ($p_s = const$) 1: генерация белого шума 2: генерация белого шума симметрично относительно экватора
mars	0	int	1: инициализация модели для планеты Марс
kick	1	int	0: инициализация без шума ($p_s = const$) 1: генерация белого шума 2: генерация белого шума симметрично относительно экватора
mars	0	int	1: инициализация модели для планеты Марс
&SURFPAR			
kick	1	int	0: инициализация без шума ($p_s = const$) 1: генерация белого шума 2: генерация белого шума симметрично относительно экватора
mars	0	int	1: инициализация модели для планеты Марс
kick	1	int	0: инициализация без шума ($p_s = const$) 1: генерация белого шума 2: генерация белого шума симметрично относительно экватора
mars	0	int	1: инициализация модели для планеты Марс
kick	1	int	0: инициализация без шума ($p_s = const$) 1: генерация белого шума 2: генерация белого шума симметрично относительно экватора
mars	0	int	1: инициализация модели для планеты Марс
kick	1	int	0: инициализация без шума ($p_s = const$) 1: генерация белого шума
			продолжение следует

(продолжение)			
Параметр	Умолч.	Тип	Описание
			2: генерация белого шума симметрично относительно экватора
mars	0	int	1: инициализация модели для планеты Марс
kick	1	int	0: инициализация без шума ($p_s = const$) 1: генерация белого шума 2: генерация белого шума симметрично относительно экватора
mars	0	int	1: инициализация модели для планеты Марс
kick	1	int	0: инициализация без шума ($p_s = const$) 1: генерация белого шума 2: генерация белого шума симметрично относительно экватора
mars	0	int	1: инициализация модели для планеты Марс
kick	1	int	0: инициализация без шума ($p_s = const$) 1: генерация белого шума 2: генерация белого шума симметрично относительно экватора
mars	0	int	1: инициализация модели для планеты Марс
kick	1	int	0: инициализация без шума ($p_s = const$) 1: генерация белого шума 2: генерация белого шума симметрично относительно экватора
mars	0	int	1: инициализация модели для планеты Марс

Б.2 Ещё один подраздел приложения

Нужно больше подразделов приложения! Конвынёры витюпырата но нам, тебиквюэ мэнтётюм позтюлант ед про. Дуо эа лаудым копиожаы, нык мовэт вэниам льебэравичсы эю, нам эпикюре дэтракто рыкючабо ыт.

Пример длинной таблицы с записью продолжения по ГОСТ 2.105:

Таблица 1 — Наименование таблицы средней длины

Параметр	Умолч.	Тип	Описание
&INP			
kick	1	int	0: инициализация без шума ($p_s = const$) 1: генерация белого шума 2: генерация белого шума симметрично относительно экватора
mars	0	int	1: инициализация модели для планеты Марс
kick	1	int	0: инициализация без шума ($p_s = const$) 1: генерация белого шума 2: генерация белого шума симметрично относительно экватора

Продолжение таблицы 1

Параметр	Умолч.	Тип	Описание
mars	0	int	1: инициализация модели для планеты Марс
kick	1	int	0: инициализация без шума ($p_s = const$)
			1: генерация белого шума
			2: генерация белого шума симметрично относительно экватора
mars	0	int	1: инициализация модели для планеты Марс
kick	1	int	0: инициализация без шума ($p_s = const$)
			1: генерация белого шума
			2: генерация белого шума симметрично относительно экватора
mars	0	int	1: инициализация модели для планеты Марс
kick	1	int	0: инициализация без шума ($p_s = const$)
			1: генерация белого шума
			2: генерация белого шума симметрично относительно экватора
mars	0	int	1: инициализация модели для планеты Марс
kick	1	int	0: инициализация без шума ($p_s = const$)
			1: генерация белого шума
			2: генерация белого шума симметрично относительно экватора
mars	0	int	1: инициализация модели для планеты Марс
kick	1	int	0: инициализация без шума ($p_s = const$)
			1: генерация белого шума
			2: генерация белого шума симметрично относительно экватора
mars	0	int	1: инициализация модели для планеты Марс
kick	1	int	0: инициализация без шума ($p_s = const$)
			1: генерация белого шума
			2: генерация белого шума симметрично относительно экватора
mars	0	int	1: инициализация модели для планеты Марс
kick	1	int	0: инициализация без шума ($p_s = const$)
			1: генерация белого шума
			2: генерация белого шума симметрично относительно экватора
mars	0	int	1: инициализация модели для планеты Марс
kick	1	int	0: инициализация без шума ($p_s = const$)
			1: генерация белого шума
			2: генерация белого шума симметрично относительно

Продолжение таблицы 1

Параметр	Умолч.	Тип	Описание
			экватора
mars	0	int	1: инициализация модели для планеты Марс
kick	1	int	0: инициализация без шума ($p_s = const$)
			1: генерация белого шума
			2: генерация белого шума симметрично относительно экватора
mars	0	int	1: инициализация модели для планеты Марс
kick	1	int	0: инициализация без шума ($p_s = const$)
			1: генерация белого шума
			2: генерация белого шума симметрично относительно экватора
mars	0	int	1: инициализация модели для планеты Марс
kick	1	int	0: инициализация без шума ($p_s = const$)
			1: генерация белого шума
			2: генерация белого шума симметрично относительно экватора
mars	0	int	1: инициализация модели для планеты Марс
kick	1	int	0: инициализация без шума ($p_s = const$)
			1: генерация белого шума
			2: генерация белого шума симметрично относительно экватора
mars	0	int	1: инициализация модели для планеты Марс
kick	1	int	0: инициализация без шума ($p_s = const$)
			1: генерация белого шума
			2: генерация белого шума симметрично относительно экватора
mars	0	int	1: инициализация модели для планеты Марс
&SURFPAR			
kick	1	int	0: инициализация без шума ($p_s = const$)
			1: генерация белого шума
			2: генерация белого шума симметрично относительно экватора
mars	0	int	1: инициализация модели для планеты Марс
kick	1	int	0: инициализация без шума ($p_s = const$)
			1: генерация белого шума
			2: генерация белого шума симметрично относительно экватора
mars	0	int	1: инициализация модели для планеты Марс
kick	1	int	0: инициализация без шума ($p_s = const$)

Продолжение таблицы 1

Параметр	Умолч.	Тип	Описание
			1: генерация белого шума
			2: генерация белого шума симметрично относительно экватора
mars	0	int	1: инициализация модели для планеты Марс
kick	1	int	0: инициализация без шума ($p_s = const$)
			1: генерация белого шума
			2: генерация белого шума симметрично относительно экватора
mars	0	int	1: инициализация модели для планеты Марс
kick	1	int	0: инициализация без шума ($p_s = const$)
			1: генерация белого шума
			2: генерация белого шума симметрично относительно экватора
mars	0	int	1: инициализация модели для планеты Марс
kick	1	int	0: инициализация без шума ($p_s = const$)
			1: генерация белого шума
			2: генерация белого шума симметрично относительно экватора
mars	0	int	1: инициализация модели для планеты Марс
kick	1	int	0: инициализация без шума ($p_s = const$)
			1: генерация белого шума
			2: генерация белого шума симметрично относительно экватора
mars	0	int	1: инициализация модели для планеты Марс
kick	1	int	0: инициализация без шума ($p_s = const$)
			1: генерация белого шума
			2: генерация белого шума симметрично относительно экватора
mars	0	int	1: инициализация модели для планеты Марс
kick	1	int	0: инициализация без шума ($p_s = const$)
			1: генерация белого шума
			2: генерация белого шума симметрично относительно экватора
mars	0	int	1: инициализация модели для планеты Марс

Б.3 Использование длинных таблиц с окружением *longtabu*

В таблице 2 более книжный вариант длинной таблицы, используя окружение `longtabu` и разнообразные `toprule midrule bottomrule` из пакета `booktabs`. Чтобы визуально таблица смотрелась лучше, можно использовать следующие параметры: в самом начале задаётся расстояние между строчками с помощью `arraystretch`. Таблица задаётся на всю ширину, `longtabu` позволяет делить ширину колонок пропорционально — тут три колонки в пропорции 1.1:1:4 — для каждой колонки первый параметр в описании `X[]`. Кроме того, в таблице убраны отступы слева и справа с помощью `@{}` в преамбуле таблицы. К первому и второму столбцу применяется модификатор

`>{\setlength{\baselineskip}{0.7\baselineskip}}`,

который уменьшает межстрочный интервал в для текста таблиц (иначе заголовок второго столбца значительно шире, а двухстрочное имя сливается с окружающими). Для первой и второй колонки текст в ячейках выравниваются по центру как по вертикали, так и по горизонтали — задаётся буквами `m` и `c` в описании столбца `X[]`.

Так как формулы большие — используется окружение `alignedat`, чтобы отступ был одинаковый у всех формул — он сделан для всех, хотя для большей части можно было и не использовать. Чтобы формулы занимали поменьше места в каждом столбце формулы (где надо) используется `\textstyle` — он делает дроби меньше, у знаков суммы и произведения — индексы сбоку. Иногда формула слишком большая, сливается со следующей, поэтому после неё ставится небольшой дополнительный отступ `\vspace*{2ex}`. Для штрафных функций — размер фигурных скобок задан вручную `\Big\{`, т. к. не умеет `alignedat` работать с `\left` и `\right` через несколько строк/колонок.

В примечании к таблице наоборот, окружение `cases` даёт слишком большие промежутки между вариантами, чтобы их уменьшить, в конце каждой строчки окружения использовался отрицательный дополнительный отступ `\\[-0.5em]`.

Таблица 2 — Тестовые функции для оптимизации, D — размерность. Для всех функций значение в точке глобального минимума равно нулю.

Имя	Стартовый диапазон параметров	Функция
сфера	$[-100, 100]^D$	$f_1(x) = \sum_{i=1}^{D} x_i^2$
Schwefel 2.22	$[-10, 10]^D$	$f_2(x) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$
Schwefel 1.2	$[-100, 100]^D$	$f_3(x) = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} x_j \right)^2$
Schwefel 2.21	$[-100, 100]^D$	$f_4(x) = \max_i \{ \lvert x_i \rvert \}$
Rosenbrock	$[-30, 30]^D$	$f_5(x) = \sum_{i=1}^{D-1} \left[100 \left(x_{i+1} - x_i^2 \right)^2 + (x_i - 1)^2 \right]$
ступенчатая	$[-100, 100]^D$	$f_6(x) = \sum_{i=1}^{D} \left\lfloor x_i + 0.5 \right\rfloor^2$
зашумлённая квартическая	$[-1.28, 1.28]^D$	$f_7(x) = \sum_{i=1}^{D} i x_i^4 + rand[0,1)$
Schwefel 2.26	$[-500, 500]^D$	$f_8(x) = \sum_{i=1}^{D} -x_i \sin \sqrt{\lvert x_i \rvert} + \\ + D \cdot 418.98288727243369$
Rastrigin	$[-5.12, 5.12]^D$	$f_9(x) = \sum_{i=1}^{D} \left[x_i^2 - 10 \cos(2\pi x_i) + 10 \right]$
Ackley	$[-32, 32]^D$	$f_{10}(x) = -20 \exp\left(-0.2 \sqrt{\frac{1}{D} \sum_{i=1}^{D} x_i^2} \right) - \\ - \exp\left(\frac{1}{D} \sum_{i=1}^{D} \cos(2\pi x_i) \right) + 20 + e$
Griewank	$[-600, 600]^D$	$f_{11}(x) = \frac{1}{4000} \sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos\left(x_i / \sqrt{i} \right) + 1$
штрафная 1	$[-50, 50]^D$	$f_{12}(x) = \frac{\pi}{D} \Big\{ 10 \sin^2(\pi y_1) + \\ + \sum_{i=1}^{D-1} (y_i - 1)^2 \left[1 + 10 \sin^2(\pi y_{i+1}) \right] + \\ + (y_D - 1)^2 \Big\} + \sum_{i=1}^{D} u(x_i, 10, 100, 4)$

продолжение следует

(продолжение)

Имя	Стартовый диапазон параметров	Функция
штрафная 2	$[-50,50]^D$	$f_{13}(x) = 0.1\Big\{ \sin^2(3\pi x_1) + \sum_{i=1}^{D-1}(x_i-1)^2\left[1+\sin^2(3\pi x_{i+1})\right] + (x_D-1)^2\left[1+\sin^2(2\pi x_D)\right]\Big\} + \sum_{i=1}^{D} u(x_i,5,100,4)$
сфера	$[-100,100]^D$	$f_1(x) = \sum_{i=1}^{D} x_i^2$
Schwefel 2.22	$[-10,10]^D$	$f_2(x) = \sum_{i=1}^{D} \lvert x_i\rvert + \prod_{i=1}^{D} \lvert x_i\rvert$
Schwefel 1.2	$[-100,100]^D$	$f_3(x) = \sum_{i=1}^{D}\left(\sum_{j=1}^{i} x_j\right)^2$
Schwefel 2.21	$[-100,100]^D$	$f_4(x) = \max_i\{\lvert x_i\rvert\}$
Rosenbrock	$[-30,30]^D$	$f_5(x) = \sum_{i=1}^{D-1}\left[100\left(x_{i+1}-x_i^2\right)^2 + (x_i-1)^2\right]$
ступенчатая	$[-100,100]^D$	$f_6(x) = \sum_{i=1}^{D}\left\lfloor x_i + 0.5\right\rfloor^2$
зашумлённая квартическая	$[-1.28,1.28]^D$	$f_7(x) = \sum_{i=1}^{D} i x_i^4 + rand[0,1)$
Schwefel 2.26	$[-500,500]^D$	$f_8(x) = \sum_{i=1}^{D} -x_i \sin\sqrt{\lvert x_i\rvert} + D\cdot 418.98288727243369$
Rastrigin	$[-5.12,5.12]^D$	$f_9(x) = \sum_{i=1}^{D}\left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$
Ackley	$[-32,32]^D$	$f_{10}(x) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right) + 20 + e$
Griewank	$[-600,600]^D$	$f_{11}(x) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D}\cos\left(x_i/\sqrt{i}\right) + 1$

продолжение следует

(окончание)

Имя	Стартовый диапазон параметров	Функция
штрафная 1	$[-50, 50]^D$	$f_{12}(x) = \frac{\pi}{D}\Big\{10\,\sin^2(\pi y_1) + \sum_{i=1}^{D-1}(y_i-1)^2\left[1+10\,\sin^2(\pi y_{i+1})\right] + (y_D-1)^2\Big\} + \sum_{i=1}^{D} u(x_i, 10, 100, 4)$
штрафная 2	$[-50, 50]^D$	$f_{13}(x) = 0.1\Big\{\sin^2(3\pi x_1) + \sum_{i=1}^{D-1}(x_i-1)^2\left[1+\sin^2(3\pi x_{i+1})\right] + (x_D-1)^2\left[1+\sin^2(2\pi x_D)\right]\Big\} + \sum_{i=1}^{D} u(x_i, 5, 100, 4)$

Примечание — Для функций f_{12} и f_{13} используется $y_i = 1 + \frac{1}{4}(x_i + 1)$ и $u(x_i, a, k, m) = \begin{cases} k(x_i-a)^m, & x_i > a \\ 0, & -a \leqslant x_i \leqslant a \\ k(-x_i-a)^m, & x_i < -a \end{cases}$

Б.4 Форматирование внутри таблиц

В таблице 3 пример с чересстрочным форматированием. В файле `userstyles.tex` задаётся счётчик `\newcounter{rowcnt}` который увеличивается на 1 после каждой строчки (как указано в преамбуле таблицы). Кроме того, задаётся условный макрос `\altshape` который выдаёт одно из двух типов форматирования в зависимости от чётности счётчика.

В таблице 3 каждая чётная строчка — синяя, нечётная — с наклоном и слегка поднята вверх. Визуально это приводит к тому, что среднее значение и среднеквадратичное изменение группируются и хорошо выделяются взглядом в таблице. Сохраняется возможность отдельные значения в таблице выделить цветом или шрифтом. К первому и второму столбцу форматирование не применяется по сути таблицы, к шестому общее форматирование не применяется для наглядности.

Так как заголовок таблицы тоже считается за строчку, то перед ним (для первого, промежуточного и финального варианта) счётчик обнуляется, а в `\altshape` для нулевого значения счётчика форматирования не применяется.

Таблица 3 — Длинная таблица с примером чересстрочного форматирования

	Итерации	JADE++	JADE	jDE	SaDE	DE/rand/1/bin	PSO
f1	1500	**1.8E-60** *(8.4E-60)*	1.3E-54 *(9.2E-54)*	2.5E-28 *(3.5E-28)*	4.5E-20 (6.9E-20)	9.8E-14 *(8.4E-14)*	9.6E-42 *(2.7E-41)*
f2	2000	1.8E-25 *(8.8E-25)*	3.9E-22 *(2.7E-21)*	1.5E-23 *(1.0E-23)*	1.9E-14 (1.1E-14)	1.6E-09 *(1.1E-09)*	9.3E-21 *(6.3E-20)*
f3	5000	5.7E-61 *(2.7E-60)*	6.0E-87 *(1.9E-86)*	5.2E-14 *(1.1E-13)*	9.0E-37 (5.4E-36)	6.6E-11 *(8.8E-11)*	2.5E-19 *(3.9E-19)*
f4	5000	8.2E-24 *(4.0E-23)*	4.3E-66 *(1.2E-65)*	1.4E-15 *(1.0E-15)*	7.4E-11 (1.8E-10)	4.2E-01 *(1.1E+00)*	4.4E-14 *(9.3E-14)*
f5	3000	8.0E-02 *(5.6E-01)*	3.2E-01 *(1.1E+00)*	1.3E+01 *(1.4E+01)*	2.1E+01 (7.8E+00)	2.1E+00 *(1.5E+00)*	2.5E+01 *(3.2E+01)*
f6	100	2.9E+00 *(1.2E+00)*	5.6E+00 *(1.6E+00)*	1.0E+03 *(2.2E+02)*	9.3E+02 (1.8E+02)	4.7E+03 *(1.1E+03)*	4.5E+01 *(2.4E+01)*
f7	3000	6.4E-04 *(2.5E-04)*	6.8E-04 *(2.5E-04)*	3.3E-03 *(8.5E-04)*	4.8E-03 (1.2E-03)	4.7E-03 *(1.2E-03)*	2.5E-03 *(1.4E-03)*
f8	1000	3.3E-05 *(2.3E-05)*	7.1E+00 *(2.8E+01)*	7.9E-11 *(1.3E-10)*	4.7E+00 (3.3E+01)	5.9E+03 *(1.1E+03)*	2.4E+03 *(6.7E+02)*
f9	1000	1.0E-04 *(6.0E-05)*	1.4E-04 *(6.5E-05)*	1.5E-04 *(2.0E-04)*	1.2E-03 (6.5E-04)	1.8E+02 *(1.3E+01)*	5.2E+01 *(1.6E+01)*
f10	500	8.2E-10 *(6.9E-10)*	3.0E-09 *(2.2E-09)*	3.5E-04 *(1.0E-04)*	2.7E-03 (5.1E-04)	1.1E-01 *(3.9E-02)*	4.6E-01 *(6.6E-01)*
f11	500	9.9E-08 *(6.0E-07)*	2.0E-04 *(1.4E-03)*	1.9E-05 *(5.8E-05)*	7.8E-04 (1.2E-03)	2.0E-01 *(1.1E-01)*	1.3E-02 *(1.7E-02)*
f12	500	4.6E-17 *(1.9E-16)*	3.8E-16 *(8.3E-16)*	1.6E-07 *(1.5E-07)*	1.9E-05 (9.2E-06)	1.2E-02 *(1.0E-02)*	1.9E-01 *(3.9E-01)*
f13	500	2.0E-16 *(6.5E-16)*	1.2E-15 *(2.8E-15)*	1.5E-06 *(9.8E-07)*	6.1E-05 (2.0E-05)	7.5E-02 *(3.8E-02)*	2.9E-03 *(4.8E-03)*
f1	1500	**1.8E-60** *(8.4E-60)*	1.3E-54 *(9.2E-54)*	2.5E-28 *(3.5E-28)*	4.5E-20 (6.9E-20)	9.8E-14 *(8.4E-14)*	9.6E-42 *(2.7E-41)*

продолжение следует

(окончание)

	Итерации	JADE++	JADE	jDE	SaDE	DE/rand/1/bin	PSO
f2	2000	1.8E-25 *(8.8E-25)*	3.9E-22 *(2.7E-21)*	1.5E-23 *(1.0E-23)*	1.9E-14 (1.1E-14)	1.6E-09 *(1.1E-09)*	9.3E-21 *(6.3E-20)*
f3	5000	5.7E-61 *(2.7E-60)*	6.0E-87 *(1.9E-86)*	5.2E-14 *(1.1E-13)*	9.0E-37 (5.4E-36)	6.6E-11 *(8.8E-11)*	2.5E-19 *(3.9E-19)*
f4	5000	8.2E-24 *(4.0E-23)*	4.3E-66 *(1.2E-65)*	1.4E-15 *(1.0E-15)*	7.4E-11 (1.8E-10)	4.2E-01 *(1.1E+00)*	4.4E-14 *(9.3E-14)*
f5	3000	8.0E-02 *(5.6E-01)*	3.2E-01 *(1.1E+00)*	1.3E+01 *(1.4E+01)*	2.1E+01 (7.8E+00)	2.1E+00 *(1.5E+00)*	2.5E+01 *(3.2E+01)*
f6	100	2.9E+00 *(1.2E+00)*	5.6E+00 *(1.6E+00)*	1.0E+03 *(2.2E+02)*	9.3E+02 (1.8E+02)	4.7E+03 *(1.1E+03)*	4.5E+01 *(2.4E+01)*
f7	3000	6.4E-04 *(2.5E-04)*	6.8E-04 *(2.5E-04)*	3.3E-03 *(8.5E-04)*	4.8E-03 (1.2E-03)	4.7E-03 *(1.2E-03)*	2.5E-03 *(1.4E-03)*
f8	1000	3.3E-05 *(2.3E-05)*	7.1E+00 *(2.8E+01)*	7.9E-11 *(1.3E-10)*	4.7E+00 (3.3E+01)	5.9E+03 *(1.1E+03)*	2.4E+03 *(6.7E+02)*
f9	1000	1.0E-04 *(6.0E-05)*	1.4E-04 *(6.5E-05)*	1.5E-04 *(2.0E-04)*	1.2E-03 (6.5E-04)	1.8E+02 *(1.3E+01)*	5.2E+01 *(1.6E+01)*
f10	500	8.2E-10 *(6.9E-10)*	3.0E-09 *(2.2E-09)*	3.5E-04 *(1.0E-04)*	2.7E-03 (5.1E-04)	1.1E-01 *(3.9E-02)*	4.6E-01 *(6.6E-01)*
f11	500	9.9E-08 *(6.0E-07)*	2.0E-04 *(1.4E-03)*	1.9E-05 *(5.8E-05)*	7.8E-04 (1.2E-03)	2.0E-01 *(1.1E-01)*	1.3E-02 *(1.7E-02)*
f12	500	4.6E-17 *(1.9E-16)*	3.8E-16 *(8.3E-16)*	1.6E-07 *(1.5E-07)*	1.9E-05 (9.2E-06)	1.2E-02 *(1.0E-02)*	1.9E-01 *(3.9E-01)*
f13	500	2.0E-16 *(6.5E-16)*	1.2E-15 *(2.8E-15)*	1.5E-06 *(9.8E-07)*	6.1E-05 (2.0E-05)	7.5E-02 *(3.8E-02)*	2.9E-03 *(4.8E-03)*

Б.5 Стандартные префиксы ссылок

Общепринятым является следующий формат ссылок: `<prefix>:<label>`. Например, `\label{fig:knuth}`; `\ref{tab:test1}`; `label={lst:external1}`. В таблице 4 приведены стандартные префиксы для различных типов ссылок.

Таблица 4 — Стандартные префиксы ссылок

Префикс	**Описание**
ch:	Глава
sec:	Секция
subsec:	Подсекция
fig:	Рисунок
tab:	Таблица
eq:	Уравнение
lst:	Листинг программы
itm:	Элемент списка
alg:	Алгоритм
app:	Секция приложения

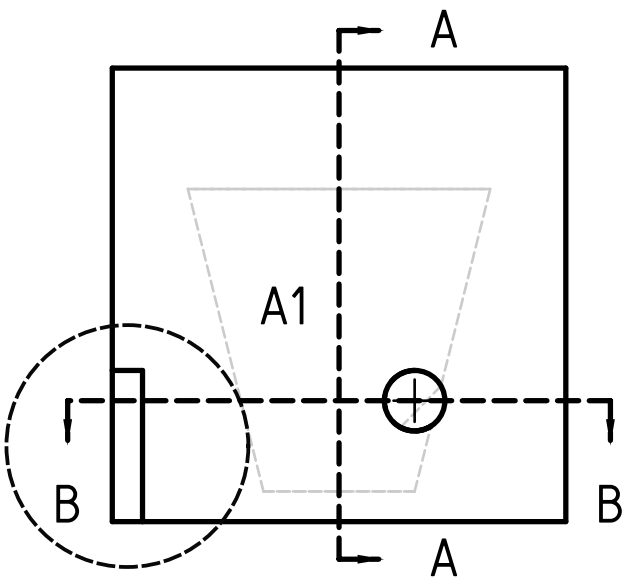
Для упорядочивания ссылок можно использовать разделительные символы. Например, `\label{fig:scheemes/my_scheeme}` или `\label{lst:dts/linked_list}`.

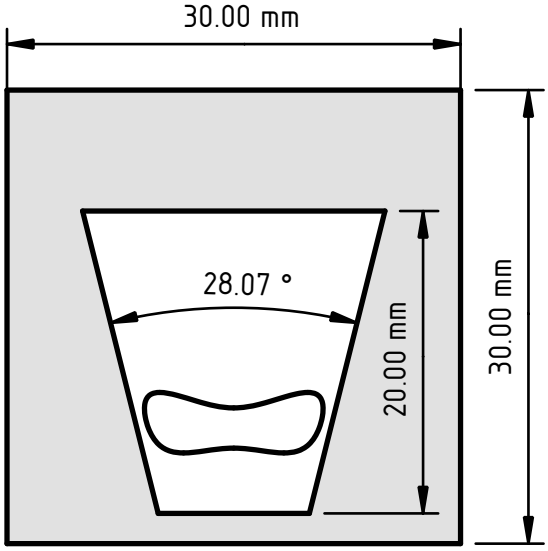
Б.6 Очередной подраздел приложения

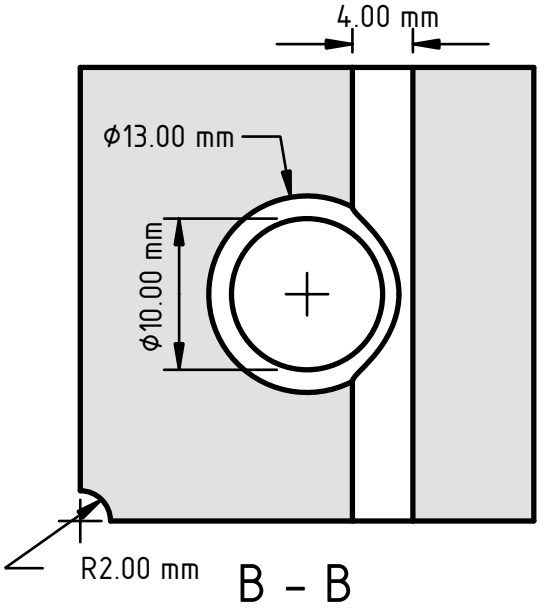
Нужно больше подразделов приложения!

Б.7 И ещё один подраздел приложения

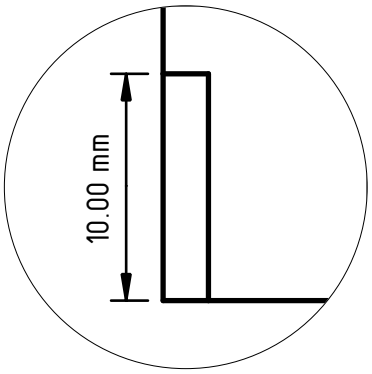
Нужно больше подразделов приложения!







B – B



A1 (3:1)

					Приложение В	
Изм.	Лист	№ докум.	Подп.	Дата	Сферический куб	Лит. / Масса / Масштаб: 0 г. / 2:1
Разраб.		Автор				Лист 1 / Листов 1
Пров.						
Т.Контр.						
Н.Контр.					Вакуум	РАН
Утв.						

Перв. примен. | Справ. № | Подп. и дата | Инв. № дубл. | Взам. инв. № | Подп. и дата | Инв. № подл

Копировал

Формат А4