Санкт-Петербургский государственный университет

Программная инженерия

Системное программирование

Сусанина Юлия Алексеевна

От синтаксического анализа графов к системам матричных уравнений

Отчёт по научно-исследовательской работе

Научный руководитель:
к. ф.-м. н., доцент Григорьев С. В.

Санкт-Петербург
2020

Оглавление

Введение

Поиск путей с контекстно-свободными ограничениями в графе (синтаксический анализ графа) — способ задать пути в помеченном графе в терминах некоторой контекстно-свободной грамматики — набирает популярность во многих областях, например, биоинформатика [19], графовые базы данных [8, 13, 20] или статический анализ кода [9, 15]. Контекстно-свободные грамматики обладают большей выразительной мощностью, чем обычно использующиеся регулярные языки и поэтому является многообещающей областью для исследования.

Несмотря на то, что уже предложено много различных алгоритмов для синтаксического анализа графов [2, 6, 13, 14, 17], недавние эксперименты показывают, что хорошая производительность на реальных данных все еще является большой проблемой [8]. Лучшие результаты показывают алгоритмы, в основе которых лежит сведение к какой-либо хорошо исследованной задаче с известными быстрыми алгоритмами решения. Например, предложенный Рустамом Азимовым алгоритм [2], сводит задачу поиска путей с контекстно-свободными ограничениями к перемножению булевых матриц и позволяет ускорить вычисления за счет использования высокопроизводительных библиотек и параллельных техник [7].

Недавние результаты применения вычислительных методов в логическом программировании показали, что использование знаний и методов из этого раздела математики позволяет ускорить вычисления [1, 16], например, сведение к решению линейных матричных уравнений ускорило вычисление определенного класса Datalog-запросов. Если мы применим похожую технику к нашей задаче, это откроет новые возможности для поиска путей с контекстно-свободными ограничениями. Например, приближенные методы часто сокращают процесс вычисления без потери важных результатов. Еще одной немаловажной причиной для выбора численных методов как перспективной области исследования является связанное с активным развитием искусственного интеллекта и машинного обучения постоянное улучшение библиотек и программных пакетов для вычислительной математики и, соответственно, возможность высокоэффективных реализаций.

В данной работе предполагается создать подход, в основе которого лежат алгоритмы линейной алгебры и вычислительной математики, который позволит ускорить обработку контектсно-свободных запросов к графовым данным. Также необходимо изучить возможность решения некоторых задач из линейной алгебры через синтаксический анализ графов, наметить основные направления будущих исследований.

1. Постановка задачи

Целью данной работы является создание подхода, основанного на методах линейной алгебры и вычислительной математики, к задаче поиска путей с контекстно-свободными ограничениями в графе. Для её достижения были поставлены следующие задачи.

- Сведение задачи синтаксического анализа графов к задаче решения (систем) матричных уравнений.
- Исследование способов решения полученных уравнений.
- Реализация и экспериментальное исследование.

2. Обзор предметной области

2.1. Синтаксический анализ

Алфавитом Σ будем называть некоторое конечное множество символов. Тогда Σ^* — это множество всех конечных строк над алфавитом Σ.

Контекстно-свободная (КС) грамматика — это четверка (Σ, N, P, S), где Σ — конечное множетство терминальных символов, N — конечное множество нетерминальных символов, P — конечное множество правил вида $A \to \beta$, где $A \in N$, $\beta \in V^*$, $V = \Sigma \cup N$ и $S \in N$ — стартовый символ.

КС-грамматика $G_S = (\Sigma, N, P, S)$ находится в нормальной форме Хомского, если все ее правила имеют следующий вид: $A \to BC$, $A \to a$, или $S \to \varepsilon$, где $A, B, C \in N, a \in \Sigma$ и ε — пустая строка.

$L_G(A) = \{\omega | A \xrightarrow{*} \omega\}$ — язык, порождаемый грамматикой $G_A = (\Sigma, N, P, A)$, где $A \xrightarrow{*} \omega$ означает, что ω может быть получена из нетерминала A путем применения некотороый последовательности правил из P.

Основной задачей синтаксического анализа является определение принадлежности некоторой строки языку, заданному грамматикой.

2.2. Задача поиска путей с контекстно-свободными ограничениями в графе

Задача поиска путей с контекстно-свободными ограничениями в графе, или, далее для краткости мы ее также будем называть задача синтаксического анализа графов, — это задача поиска путей в помеченном графе, которые удовлетворяют ограничениям, заданным некоторых формальным языком, в нашем случае, контекстно-свободным.

Ориентированный, помеченный граф — это тройка $D = (V, E, \Sigma)$, где V — конечное множество вершин, $E \subseteq V \times \Sigma \times V$ — конечное множество ребер с метками из Σ.

Путь из вершины m в вершину n в графе D будем обозначать как $m\lambda n$, где λ — уникальное слово, составленное путем конкатенации всех меток ребер вдоль данного пути. Обозначим $\mathcal{P}$ множество всевозможных путей в графе D.

Тогда задача поиска путей с контекстно-свободными ограничениями в терминах реляционной семантики запросов [11] — это для входных КС-грамматики G и помеченного графа D найти контекстно-свободные отношения $R_A \subseteq V \times V$ для каждого нетерминала $A \in N$, которые заданы следующим образом: $R_A = \{(m, n) \mid m\lambda n \in \mathcal{P}, \lambda \in L_G(A)\}$. Так как каждое отношение R_A конечно, то его можно представить с помощью Булевой матрицы T_A: $T_A = 1 \Leftrightarrow (i, j) \in R_A$.

2.3. Матричный алгоритм для синтаксического анализа графов

Матричный алгоритм был предложен Рустамом Азимовым [2] и вычисляет контектсно-свободные запросы с использованием реляционной семантики. Он строит матрицу разбора T, элементами которой являются множетсва нетерминалов, такие что $A \in T_{i,j} \Leftrightarrow \exists i \lambda j \in R_A$.

Для каждой пары вершин (i, j) инициализируем $T_{i,j} = \{A \mid (i, a, j) \in E, (A \to a) \in P\}$. Затем, заполнение матрицы разбора T происходит путем вычисление ее транзитивного замыкания: $M^* = M^{(1)} \cup M^{(2)} \ldots$, где $M^{(1)} = M$, $M^{(k)} = M^{(k-1)} \cup (M^{(k-1)} \times M^{(k-1)})$ for $k > 1$.

Также мы можем представить матрицу T как множество Булевых матриц размера $|V| \times |V|$, такие что $(T_A)_{i,j} = 1 \Leftrightarrow A \in T_{i,j}$ для каждого нетерминала $A \in N$. Таким образом, мы можем заменить вычисление транзитивного замыкания $T = T \cup (T \times T)$ на несколько умножений Булевых матриц $T_A = T_A + (T_B T_C)$ для каждого правила в грамматике $(A \to BC) \in P$. Псевдокод данного алгоритма приведен в Листинге 1.

Листинг 1 Матричный алгоритм для синтаксического анализа графов

```
1: T ← the matrix |V| × |V| in which each element is ∅
2: for all (i, x, j) ∈ E do
3:     T_{i,j} ← T_{i,j} ∪ {A | (A → x) ∈ P}
4: while matrix T is changing do
5:     T ← T ∪ (T × T)
6: return T
```

Несмотря на то, что сложность данного алгоритма $\mathcal{O}(|N|^3|V|^2(BMM(|V|) + BMU(|V|)))$ (здесь $BMM(n)$ и $BMU(n)$ обозначают время, необходимое для перемножения и сложения двух Булевых матриц размера $n \times n$ соответственно), его реализации, в которых используются техники параллельных вычислений и эффективные алгоритмы обработки матричных операций, показывают высокую производительность на реальных данных [7].

2.4. Datalog-запросы и синтаксический анализ графов

Datalog — декларативный логический язык программирования, использующийся для написания и вычисления запросов к базам данных [5].

Программа на языке Datalog состоит из базы данных фактов, которую также называют экстенсиональной базой данных, и правил, с помощью которых можно выводить новые факты из уже имеющихся. Правила предназначены для построения интенсиональной базы данных.

Рассмотрим простую грамматику для описания вложенных скобочных последовательностей: $S \to aSb \mid ab$ и произвольный граф $D = (V, E, \{a, b\})$. Примерами фактов в данном случае являются $a(0, 1)$ или $b(1, 2)$, которые обозначают "существует ребро из вершины 0 в вершину 1 с меткой a в графе D" и "существует ребро из вершины 1 в вершину 2 с меткой b в графе D" соответсвенно. Правила будут иметь следующий вид:

$$S(i, j) :- a(i, k), S(k, l), b(l, j)$$
$$S(i, j) :- a(i, k), b(k, j).$$

А запросом будет выражение, которое вернет все такие пары (i, j), что существует путь из вершины i в вершину j, удовлетворяющие отношению R_S: $? :- S(i, j)$.

Таким образом, можно описывать КС-запросы к графам в терминах Datalog-запросов.

2.5. Вычисление Datalog-запросов через решение систем матричных уравнений

Тайсуке Сато в своей работе [16] предложил фундаментально новый подход к вычислению Datalog-запросов. Он рассматривает линейные Datalog-программы, в которых используется N констант и M бинарных предикатов. (В нашем случае, константами являются номера вершин, бинарными предикатами терминальные и нетерминальные символы.) Далее, закодировав все константы N-мерными векторами, а предикаты матрицами размера $N \times N$, он свел задачу к решению систем линейных матричных уравнений над $\mathbb{R}$. Проведенные Сато эксперименты показали, что предложенный подход эффективно справляется с Datalog-запросами к графам размера до 10^4 вершин и показывает ускорение в $10^1 \sim 10^4$ раз по сравнению с системами B-Prolog, XSB, DLV и Clingo, когда данные не являются разреженными.

Так как мы можем описывать контекстно-свободные запросы с помощью Datalog-запросов, как было показано в предыдущем подразделе, то мы можем попытаться адаптировать предложенный Сато подход к решению проблемы синтаксического анализа графов и исследовать его эффективность уже применительно к нашей задаче.

2.6. Безматричные методы Ньютона-Крылова

Поскольку, в дальнейшем, мы будем работать как с линейными, так и с нелинейными матричными уравнениями, то в данном разделе необходимо рассмотреть один из классов методов решения нелинейных матричных уравнений и его преимущества.

Пусть дана некоторая нелинейная функция $F(X) = 0$ и начальное приближение X_0, тогда метод Ньютона пытается найти решение данной функции путем численного

итеративного процесса:

$$X_{i+1} = X_i - (F'(X_i))^{-1} F(X_i)$$

или

$$\begin{cases} F'(X_i) H_i = -F(X_i) \\ X_{i+1} = X_i + H_i. \end{cases}$$

Явным преимуществом данного метода является то, что он обладает квадратичной сходимостью и позволяет находить решение за небольшое число шагов.

Из второй записи итеративного процесса видно, что на каждом шаге необходимо решать линейной уравнение $F'(X_i)H_i = -F(X_i)$. Наиболее эффективными методами решения больших несимметричных систем уравнений являются методы, основанные на подпространствах Крылова, например, метод минимизации невязок с рестартами (GMRES) и его модификации.

Эти методы используют для решения системы уравнений $Ax = b$ подпространства Крылова: $K_j = span(r_0, Ar_0, A^2 r_0, \cdot, A^{j-1} r_0)$, где $r_0 = b - Ax_0$, x_0 — некоторое начальное приближение.

Можно заметить, что при использовании методов Крылова для решения линейных систем уравнений производная функции $F'(X_i)$ встречается только в умножении вида матрица-вектор и может быть вычислена приближенно:

$$F'(X_i)v \approx \frac{F(X_i + \epsilon v) - F(X_i)}{\epsilon},$$

где ϵ — некоторое малое число.

Такие методы, основанные на методе Ньютона с использованием методов решения линейных систем с помощью подпространств Крылова и аппроксимации производной функции, называются безматричными методами Ньютона-Крылова [12].

3. Синтаксический анализ графов через решение систем матричных уравнений

В данном разделе мы собираемся на основе метода, предложенного Сато, свести задачу синтаксического анализа графов к задаче решения систем матричных уравнений. Предполагается, что данное сведение в будущем позволит создавать эффективные реализации для ускорения вычисления контекстно-свободных запросов к графам. Также Сато замечает, что решение нелинейных матричных уравнений точными методами вызывает затруднения как с точки зрения теории, так и с точки зрения практики из-за возникающих проблем с точностью.

Сначала сведем задачу поиска путей с контекстно-свободными ограничениями в графе к решению систем булевых матричных уравнений.

Для данных графа D и грамматики G каждый нетерминал и терминал этой грамматики задает конечное отношение R_A, которое может быть представлено как Булева матрица аналогично с представлением с матричном алгоритме:

$$T_E \in \mathbb{M}^{|V|\times|V|} : (T_E)_{ij} = 1 \iff (i,j) \in R_E \ \ \forall E \in (N \cup \Sigma).$$

Теперь для каждого правила:

$$N_i \to \beta_0^0 \dots \beta_k^0 \mid \dots \mid \beta_0^l \dots \beta_m^l, \beta_j^i \in \Sigma \cup N$$

мы можем создать соответсвующее ему уравнение над Булевыми матрицами

$$T_{N_i} = T_{\beta_0^0} \cdot \ldots \cdot T_{\beta_k^0} + \ldots + T_{\beta_0^l} \cdot \ldots \cdot T_{\beta_m^l}.$$

Например, рассмотрим небольшой запрос $a^n b^n$, который не может быть описан с помощью регулярного языка. Контекстно-свободная грамматика для него выглядит следующим образом: $S \to aSb \mid ab$. Данный запрос может быть преобразован в следующее уравнение над Булевыми матрицами:

$$T_S = T_A T_S T_B + T_A T_B. \tag{1}$$

Один из способов его решения — наивный итеративный процесс:

$$T_S^0 = \mathbf{0}$$
$$T_S^{k+1} = T_A T_S^k T_B + T_A T_B$$

Но, чтобы применять эффективные численные методы, мы должны рассматреть другое уравнение над вещественными числами $\mathbb{R}$:

$$\mathcal{T}_S = \epsilon(T_A \mathcal{T}_S T_B + T_A T_B).$$

Для этого уравнения мы также строим последовательность матриц $\{\mathcal{T}_S^k\}$:

$$\mathcal{T}_S^0 = \mathbf{0}$$
$$\mathcal{T}_S^{k+1} = \epsilon(T_A \mathcal{T}_S^k T_B + T_A T_B)$$

Когда $\mathcal{T}_S^k \leq \mathbf{1}$ данная последовательность сходится к $\mathcal{T}_S^*$, так как является монотонно возрастающей последовательностью матриц с точной верней границей. Более того, $((\mathcal{T}_S^{k+1})_{ij} > 0 \Leftrightarrow (T_S^{k+1})_{ij} = 1)$ и, следовательно, $ceil(\mathcal{T}_S^*) = T_S^*$, где $ceil(x)$ — функция, возвращающая наименьшее целое, которое меньше или равно x. В следующем разделе будет показано, почему это так и как подбирать необходимую ϵ.

Таким образом, для каждого правила входной грамматики

$$N_i \to \beta_0^0 \dots \beta_k^0 \mid \dots \mid \beta_0^l \dots \beta_m^l$$

можно построить уравнение над вещественными числами

$$\mathcal{T}_{N_i} = \epsilon_{N_i}(T_{\beta_0^0} \cdot \dots \cdot T_{\beta_k^0} + \dots + T_{\beta_0^l} \cdot \dots \cdot T_{\beta_m^l}),$$

где ϵ_{N_i} выбрана так, чтобы $\mathcal{T}_{N_i}^k \leq \mathbf{1}$ для всех k в соответствующем итеративном процессе.

То есть мы показали, как можно свести решение задачи синтаксического анализа к задаче решения систем матричных уравнений. Теперь необходимо показать, почему такое сведение является корректным и как эффективно такие системы решать.

4. Исследование методов решения полученных уравнений

Здесь мы попробуем рассмотреть способы решения полученных уравнений. Для наглядности, как и в предыдущем разделе, мы остановимся на небольших грамматиках, в которых один нетерминал. Однако при работе с реальными данными приходится иметь дело с большими грамматиками, в которых более одного нетерминала, поэтому мы также кратко опишем методы решения в тех, случаях, когда мы сталкиваемся с системами уравнений.

4.1. Линейный случай

Рассмотрим уравнение для грамматики $S \to aSb \mid ab$, полученное в предыдущем разделе:

$$\mathcal{T}_S = \varepsilon(T_A T_B + T_A \mathcal{T}_S T_B). \tag{2}$$

Это уравнение можно привести к линейной системе с помощью произведения Кронекера:

$$vec(\mathcal{T}_S) = \varepsilon(vec(T_A T_B) + (T_B^T \otimes T_A) vec(\mathcal{T}_S)),$$

где $vec(X)$ — функция соединяющая столбцы матрицы X в один единый вектор.

Заметим, что большинство доказательств дублирует ход рассуждений Сато в [16], однако мы рассматриваем контекстно-свободную грамматику, в то время как Сато рассматривал в основном регулярные.

Лемма 1. *Если* $0 < \varepsilon \leq \dfrac{1}{1 + \|T_B^T \otimes T_A\|}$, *тогда последовательность матриц* $\{\mathcal{T}_S^{(k)}\}$*:*

$$\mathcal{T}_S^{(0)} = 0$$
$$vec(\mathcal{T}_S^{(k+1)}) = \varepsilon(vec(T_A T_B) + (T_B^T \otimes T_A) vec(\mathcal{T}_S^{(k)}))$$

сходится и ее предел равен наименьшему решению уравнения 2.

Доказательство. Сначала докажем, что $\mathcal{T}_S^{(k)} \leq \mathbb{1}\ \forall k \in \mathbb{N}$.

База индукции: очевидно.

Индукционный переход: пусть $\mathcal{T}_S^{(k)} \leq \mathbb{1}$.

$$\begin{aligned} vec(\mathcal{T}_S^{(k+1)}) &= \varepsilon(vec(T_A T_B) + (T_B^T \otimes T_A) vec(\mathcal{T}_S^{(k)})) \\ &\leq \varepsilon(vec(T_A T_B) + (T_B^T \otimes T_A)\mathbb{1}) \\ &\leq \varepsilon(1 + \|T_B^T \otimes T_A\|)\mathbb{1} \end{aligned}$$

Так как $\varepsilon(1 + \|T_B^T \otimes T_A\|) \leq 1$, то $\mathcal{T}_S^{(k+1)} \leq \mathbb{1}$

$\{\mathcal{T}_S^{(k)}\}$ сходится как последовательность монотонно возрастающих матриц с существующей верхней границей

$$\begin{aligned}\mathcal{T}_S^{(\infty)} = \lim_{k\to\infty} \mathcal{T}_S^{(k+1)} &= \lim_{k\to\infty} \varepsilon(T_A T_B + T_A \mathcal{T}_S^{(k)} T_B) \\ &= \varepsilon(T_A T_B + T_A \lim_{k\to\infty} \mathcal{T}_S^{(k)} T_B) = \varepsilon(T_A T_B + T_A \mathcal{T}_S^{(\infty)} T_B)\end{aligned}$$

Теперь покажем, что для любого другого решения $\mathcal{T}'_S$ выполняется $\mathcal{T}_S^{(\infty)} \le \mathcal{T}'_S$. Для этого покажем, что $\mathcal{T}_S^{(k)} \le \mathcal{T}'_S \ \forall k \in \mathbb{N}$

База индукции: $0 \le \mathcal{T}'_S$.

Индукционный переход: пусть $\mathcal{T}_S^{(k)} \le \mathcal{T}'_S$.

$$\begin{aligned}vec(\mathcal{T}_S^{(k+1)}) &= \varepsilon(vec(T_A T_B) + (T_B^T \otimes T_A) vec(\mathcal{T}_S^{(k)})) \\ &\le \varepsilon(1 + \|T_B^T \otimes T_A\|)\mathcal{T}_S^{(k)}\end{aligned}$$

Так как $\varepsilon(1 + \|T_B^T \otimes T_A\|) \le 1$ и $\mathcal{T}_S^{(k)} \le \mathcal{T}'_S$, то $\mathcal{T}_S^{(k+1)} \le \mathcal{T}'_S$ и, следовательно,

$$\mathcal{T}_S^{(\infty)} = \lim_{k\to\infty} \mathcal{T}_S^{(k+1)} \le \mathcal{T}'_S.$$

□

Лемма 2. $(T_S^{(k)})_{i,j} = 1 \Leftrightarrow (\mathcal{T}_S^{(k)})_{i,j} > 0$

Доказательство. База индукции: очевидно.

Индукционный переход: $(T_S^{(k)})_{i,j} = 1 \Leftrightarrow (\mathcal{T}_S^{(k)})_{i,j} > 0$.

$$\begin{aligned}(T_A T_S^{(k)} T_B)_{i,j} = 1 &\Leftrightarrow \exists p,q : (T_A)_{i,p} (T_S^{(k)})_{p,q} (T_B)_{q,j} = 1 \\ &\Leftrightarrow \exists p,q : (T_A)_{i,p} = 1 \text{ и } (T_S^{(k)})_{p,q} = 1 \text{ и } (T_B)_{q,j} = 1 \\ &\text{по индукционному предложению} \\ &\Leftrightarrow \exists p,q : (T_A)_{i,p} = 1 \text{ и } (\mathcal{T}_S^{(k)})_{p,q} > 0 \text{ и } (T_B)_{q,j} = 1 \\ &\Leftrightarrow (T_A \mathcal{T}_S^{(k)} T_B)_{i,j} > 0\end{aligned}$$

$$\begin{aligned}(T_S^{(k+1)})_{i,j} = 1 &\Leftrightarrow (T_A T_B + T_A T_S^{(k)} T_B)_{i,j} = 1 \\ &\Leftrightarrow (T_A T_B)_{i,j} = 1 \text{ или } (T_A T_S^{(k)} T_B)_{i,j} = 1 \\ &\Leftrightarrow (T_A T_B)_{i,j} = 1 \text{ или } (T_A \mathcal{T}_S^{(k)} T_B)_{i,j} > 0 \\ &\Leftrightarrow (\mathcal{T}_S^{(k+1)})_{i,j} = (\varepsilon(T_A T_B + T_A \mathcal{T}_S^{(k)} T_B))_{i,j} > 0\end{aligned}$$

□

Лемма 3. *Пусть $\mathcal{T}_S^*$ — наименьшее решение уравнения 2 и $0 < \varepsilon \le \dfrac{1}{1 + \|T_B^T \otimes T_A\|}$,*

тогда T_S^*, построенная следующим образом:

$$(T_S^*)_{i,j} = \begin{cases} 1, & \text{если } (\mathcal{T}_S^*)_{i,j} > 0 \\ 0, & \text{иначе} \end{cases} \tag{3}$$

является наименьшим решением уравнения 1.

Теорема 1. *Если $0 < \varepsilon \leq \dfrac{1}{1 + \|T_B^T \otimes T_A\|}$, то $\mathcal{T}_S^* = (I \otimes I - \varepsilon(T_B^T \otimes T_A))^{-1} \varepsilon \; vec(T_A T_B)$ — наименьшее решение уравнения 2, а T_S^*, построенное как в лемме 3, является наименьшим решением уравнения 1.*

Доказательство.

$$\rho(\varepsilon(T_B^T \otimes T_A))) \leq \varepsilon \|T_B^T \otimes T_A\| \leq 1 - \varepsilon < 1$$

$\Rightarrow (I \otimes I - \varepsilon(T_B^T \otimes T_A))^{-1}$ существует и уравнение 2 имеет единственное решение, которое совпадает с $\mathcal{T}_S^*$. □

Таким образом, при корректном подборе ϵ наше уравнение будет иметь единственное решение. Уравнения вида $\sum_{i=1}^{n} A_i X B_i = C$ называются уравнениями Сильвестра и для $k = 1, 2$ могут быть решены за $\mathcal{O}(|V|^3)$ [3]. В случае $k > 2$ их можно свести с линейной системе с помощью произведения Кронекера (как было показано выше) $Ax = b$, где A — матрица размера $(|V|^2 \times |V|^2)$ и время, необходимое для нахождения решения — это $\mathcal{O}(|V|^6)$ or $\mathcal{O}(|V|^{4+2\omega})$ используя более эффективные алгоритмы перемножения матриц (здесь $\mathcal{O}(n^{2+\omega})$ — временная сложность умножения двух матриц размера $n \times n$). Использование разреженного представления матриц и приближенных методов может быть эффективно для решения уравнений такого вида [4].

Если же мы имеем дело с системами уравнений, то мы должны построить граф зависимостей для нетерминалов входной грамматики и разбить множетсво наших уравнений на непересекающиеся подмножетсва в соответсвии с сильносвязными компонентами этого графа, а затем решать систему поэтапно для каждого полученного подмножетсва.

4.2. Нелинейный случай

Рассмотрим грамматику $S \to SS \mid aSb \mid ab$, которая описывает язык правильных скобочных последовательностей, и её уравнение:

$$\mathcal{T}_S = \varepsilon(\mathcal{T}_S^2 + T_A T_B + T_A \mathcal{T}_S T_B).$$

Для решения нелинейных систем лучше всего подходят различные численные итеративные методы, однако метод наивной итерации сходится долго. Мы также можем

ввести функцию $F(\mathcal{T}_S) = \mathcal{T}_S - \varepsilon(\mathcal{T}_S^2 + T_A T_B + T_A \mathcal{T}_S T_B) = 0$ и решать ее методом Ньютона, который обладает квадратичной сходимостью.

Теперь покажем, почему безматричный метод Ньютона-Крылова подходит для решения таких систем уравнений. Кроме его известных преимуществ, таких как уже упомянутая скорость сходимости и аппроксимация производной, которая позволяет существенно ускорить вычисления, данный метод также, в нашем случае, никогда не будет изменять элементы, которые в результирующей матрицы должны быть нулевыми.

На этапе решения линейной системы $F'(X_i)H_i = -F(X_i)$ методами основанными на подпространствах Крылова начальное приближение x_0 выгодно и следует брать нулевым, то $r_0 = -F(X_i)$. Производная у нас считается приближенно по следующей формуле

$$F'(X_i)v \approx \frac{F(X_i + \epsilon v) - F(X_i)}{\epsilon},$$

и в качестве вектора у нас может служить только некоторый элемент подпространства Крылова. То есть все остальные элементы образованы только применением функции либо к X_0 (которое также следует взять нулем), либо к комбинации X_i и ϵX_j, что не может привести к изменению элементов, которые в $\mathcal{T}_S^*$ равны нулю.

Главным недостатком использования метода Ньютона и вообще решения систем нелинейных уравнений является подбор ϵ, для сходимости она должна быть менее $\frac{1}{|V|}$. Данную проблему можно решать только ad-hoc учитывая размер и разреженность данных.

5. Эксперименты

Мы проведем сравнительный анализ нашего метода с матричным алгоритмом синтаксического анализа графов и рассмотрим дальнейшие пути улучшения реализаций предложенного метода.

Главной целью данных экспериментов показать применимость предложенного алгоритма на реальных данных. Для сравнения был выбрали матричный алгоритм синтаксического анализа графов [2], так как в его основе также лежит сведение к другой проблеме с уже существующими эффективными решениями, и GLL для графов [10], который обладает лучшей асимптотической сложностью, равной $\mathcal{O}(|V|^4)$. Реализации и данные результатов экспериментов были взяты из соответсвующих статей. Названия реализаций остались прежними.

Алгоритмы сравнивались на двух запросах, правила которых представлены на рисунках 1 и 2. Графы для запросов были взять из работы [6].

$$\begin{array}{lcc} S & \to & subClassOf^{-1}\ S\ subClassOf \\ S & \to & type^{-1}\ S\ type \\ S & \to & subClassOf^{-1}\ subClassOf \\ S & \to & type^{-1}\ type \end{array}$$

Рис. 1: Правила грамматики для запроса 1.

$$\begin{array}{lcc} S & \to & subClassOf^{-1}\ S\ subClassOf \\ S & \to & subClassOf \end{array}$$

Рис. 2: Правила грамматики для запроса 2.

Уравнения для запросов 1 и 2 были сконструированы и решены на языке программирования Python с помощью библиотеки *scipy* [18]:

- в **sLSV** использовалась функция *spsolve* для решения разреженных линейных систем;
- в **JFNK** использовалась функция $optimize.newton_krylov$ для решения уравнения безматричным методом Ньютона-Крылова.

Полученые результаты показывают, что предложенный метод и его первая реализация уже показывают лучший результат, чем первые реализации сравниваемых алгоримов. Во многих случаях **JFNK** проигрывает другим реализациям, потому что в нем не используются ни параллельные вычисления, ни разреженное представление матриц.

Таким образом, можно сделать вывод, что предложенный подход показывает неплохое время работы, сравнимое с другими алгоритмами синтаксического анализа

Таблица 1: Результаты сравнения с матричным алгоритмом для запроса 1 (в мс)

Ontology	V	E	GLL	dGPU	sCPU	sGPU	sLSV	JFNK
skos	144	323	10	56	14	12	9	13
generations	129	351	19	62	20	13	9	14
travel	131	397	24	69	22	30	10	65
univ-bench	179	413	25	81	25	15	13	61
atom-primitive	291	685	255	190	92	22	32	190
biomedical-measure-primitive	341	711	261	266	113	20	50	421
foaf	256	815	39	154	48	9	26	100
people-pets	337	834	89	392	142	32	51	305
funding	778	1480	212	1410	447	36	213	1910
wine	733	2450	819	2047	797	54	410	2220
pizza	671	2604	697	1104	430	24	1710	1500

Таблица 2: Результаты сравнения с матричным алгоритмом для запроса 2 (в мс)

Ontology	V	E	GLL	dGPU	sCPU	sGPU	sLSV	JFNK
skos	144	323	1	10	2	1	7	5
generations	129	351	1	9	2	0	5	0
travel	131	397	1	31	7	10	5	51
univ-bench	179	413	11	55	15	9	8	40
atom-primitive	291	685	66	36	9	2	27	40
biomedical-measure-primitive	341	711	45	276	91	24	35	284
foaf	256	815	2	53	14	3	16	26
people-pets	337	834	3	144	38	6	49	73
funding	778	1480	23	1246	344	27	184	502
wine	733	2450	8	722	179	6	171	791
pizza	671	2604	29	943	258	23	161	334

графов и демонстрирует свою практическую применимость. Однако предложенные решения плохо маштабируются и для работы с большими данными потребуется более эффективная реализация, использующая параллельные вычисления или другие высоко-производительные библиотеки для численных методов.

6. Заключение

В данной работе были получены следующие результаты.

- Задача CFPQ была сведена к задаче решения матричных уравнений над множеством вещественных чисел $\mathbb{R}$
- Исследованы способы решения полученных линейных и нелинейных уравнений как точными методами, так и приближенными
- Поставлены первые эксперименты применения полученного подхода к CFPQ и проведен сравнительный анализ с существующими аналогами
- На основе полученных результатов написана работа "Context-Free Path Querying via Matrix Equations", которая была принята во второй очный тур ACM SIGMOD 2020 Student Research Competition.

Кроме того, мы можем определить несколько направлений будущих исследований.

- Эффективная реализация предложенного подхода с использованием специализированных библиотек и параллельных вычислений.
- Определение подклассов полиномиальных уравнений, решение которых может быть сведено к CFPQ.
- Попытка построить взаимное сведение между CFPQ и решением соответсвующих подклассов уравнений.

Список литературы

[1] Aspis Yaniv. A Linear Algebraic Approach to Logic Programming : Ph. D. thesis / Yaniv Aspis ; Imperial College London. — 2018.

[2] Azimov Rustam, Grigorev Semyon. Context-free Path Querying by Matrix Multiplication // Proceedings of the 1st ACM SIGMOD Joint International Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA). — GRADES-NDA '18. — New York, NY, USA : ACM, 2018. — P. 5:1–5:10. — Access mode: http://doi.acm.org/10.1145/3210259.3210264.

[3] Bartels R. H., Stewart G. W. Solution of the Matrix Equation AX + XB = C [F4] // Commun. ACM. — 1972. — Sep. — Vol. 15, no. 9. — P. 820–826. — Access mode: http://doi.acm.org/10.1145/361573.361582.

[4] Bouhamidi Abderrahman, Jbilou Khalide. A note on the numerical approximate solutions for generalized Sylvester matrix equations with applications // Applied Mathematics and Computation. — 2008. — 12. — Vol. 206. — P. 687–694.

[5] Ceri Stefano, Gottlob Georg, Tanca Letizia. What you always wanted to know about Datalog (and never dared to ask) // IEEE transactions on knowledge and data engineering. — 1989. — Vol. 1, no. 1. — P. 146–166.

[6] Context-free path queries on RDF graphs / X. Zhang, Z. Feng, X. Wang et al. // International Semantic Web Conference / Springer. — 2016. — P. 632–648.

[7] Evaluation of the Context-Free Path Querying Algorithm Based on Matrix Multiplication / Nikita Mishin, Iaroslav Sokolov, Egor Spirin et al. // Proceedings of the 2Nd Joint International Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA). — GRADES-NDA'19. — New York, NY, USA : ACM, 2019. — P. 12:1–12:5. — Access mode: http://doi.acm.org/10.1145/3327964.3328503.

[8] An Experimental Study of Context-Free Path Query Evaluation Methods / Jochem Kuijpers, George Fletcher, Nikolay Yakovets, Tobias Lindaaker // Proceedings of the 31st International Conference on Scientific and Statistical Database Management. — SSDBM '19. — New York, NY, USA : ACM, 2019. — P. 121–132. — Access mode: http://doi.acm.org/10.1145/3335783.3335791.

[9] Fast Algorithms for Dyck-CFL-reachability with Applications to Alias Analysis / Qirun Zhang, Michael R. Lyu, Hao Yuan, Zhendong Su // Proceedings of the 34th ACM SIGPLAN Conference on Programming Language Design and

Implementation. — PLDI '13. — New York, NY, USA : ACM, 2013. — P. 435–446. — Access mode: http://doi.acm.org/10.1145/2491956.2462159.

[10] Grigorev Semyon, Ragozina Anastasiya. Context-free Path Querying with Structural Representation of Result // Proceedings of the 13th Central & Eastern European Software Engineering Conference in Russia. — CEE-SECR '17. — New York, NY, USA : ACM, 2017. — P. 10:1–10:7. — Access mode: http://doi.acm.org/10.1145/3166094.3166104.

[11] Hellings Jelle. Path Results for Context-free Grammar Queries on Graphs // CoRR. — 2015. — Vol. abs/1502.02242. — 1502.02242.

[12] Knoll D. A., Keyes D. E. Jacobian-free Newton-Krylov Methods: A Survey of Approaches and Applications // J. Comput. Phys. — 2004. — Jan. — Vol. 193, no. 2. — P. 357–397. — Access mode: http://dx.doi.org/10.1016/j.jcp.2003.08.010.

[13] Medeiros Ciro M., Musicante Martin A., Costa Umberto S. Efficient Evaluation of Context-free Path Queries for Graph Databases // Proceedings of the 33rd Annual ACM Symposium on Applied Computing. — SAC '18. — New York, NY, USA : ACM, 2018. — P. 1230–1237. — Access mode: http://doi.acm.org/10.1145/3167132.3167265.

[14] Parser Combinators for Context-free Path Querying / Ekaterina Verbitskaia, Ilya Kirillov, Ilya Nozkin, Semyon Grigorev // Proceedings of the 9th ACM SIGPLAN International Symposium on Scala. — Scala 2018. — New York, NY, USA : ACM, 2018. — P. 13–23. — Access mode: http://doi.acm.org/10.1145/3241653.3241655.

[15] Reps Thomas. Program Analysis via Graph Reachability // Proceedings of the 1997 International Symposium on Logic Programming. — ILPS '97. — Cambridge, MA, USA : MIT Press, 1997. — P. 5–19. — Access mode: http://dl.acm.org/citation.cfm?id=271338.271343.

[16] SATO TAISUKE. A linear algebraic approach to datalog evaluation // Theory and Practice of Logic Programming. — 2017. — May. — Vol. 17, no. 3. — P. 244–265. — Access mode: https://doi.org/10.1017/s1471068417000023.

[17] Santos Fred C., Costa Umberto S., Musicante Martin A. A Bottom-Up Algorithm for Answering Context-Free Path Queries in Graph Databases // Web Engineering / Ed. by Tommi Mikkonen, Ralf Klamma, Juan Hernández. — Cham : Springer International Publishing, 2018. — P. 225–233.

[18] Jones Eric, Oliphant Travis, Peterson Pearu et al. SciPy: Open source scientific tools for Python. — 2001–2019. — [Online; accessed 5.3.2019]. Access mode: http://www.scipy.org/.

[19] Sevon Petteri, Eronen Lauri. Subgraph Queries by Context-free Grammars // Journal of Integrative Bioinformatics. — 2008. — Jun. — Vol. 5, no. 2. — Access mode: https://doi.org/10.1515/jib-2008-100.

[20] Yannakakis Mihalis. Graph-theoretic Methods in Database Theory // Proceedings of the Ninth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems. — PODS '90. — New York, NY, USA : ACM, 1990. — P. 230–242. — Access mode: http://doi.acm.org/10.1145/298514.298576.