Санкт-Петербургский государственный университет

Кафедра Системного программирования

Ершов Кирилл Максимович

Синтаксический анализ графов с помеченными вершинами и ребрами

Курсовая работа

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

Санкт-Петербург
2017

Оглавление

Введение

Помеченные графы являются удобным способом представления различных структурированных данных. Такие графы используются, например, в биоинформатике, логистике, графовых базах данных.

Иногда для представления данных с использованием графов обходятся только метками на рёбрах. Но в некоторых случаях метки на вершинах позволяют более наглядно отображать зависимости между сущностями. К примеру, в биоинформатике существует большое количество данных, содержащих взаимосвязь между генами и белками. Такие данные удобно представлять в виде графа, вершины которого помечены определенными генами и белками, а ребра показывают их отношение (например, ген кодирует белок).

Для поиска информации в помеченном графе необходимо иметь возможность выполнять запросы, задающие класс путей в графе. Пути рассматриваются как строки, состоящие из меток на рёбрах и вершинах. Тогда запрос можно представить в виде грамматики: путь удовлетворяет запросу, если он принадлежит языку, который порождает заданная грамматика. Таким образом, грамматика задаёт класс путей в графе и задача выполнения запросов сводится к задаче синтаксического анализа графа.

Существуют различные языки запросов для получения нужных путей из графа. Но многие из них позволяют задавать только регулярные запросы [1, 10, 9] или поддерживают графы с метками только на рёбрах [5]. В некоторых случаях с помощью регулярных грамматик невозможно задать нужные запросы. Например, язык $\{ww^T | w$ — строка$\}$, состоящий из строк–палиндромов, нельзя задать регулярной грамматикой. Однако запросы к графу для поиска вершин одного поколения (same generation – query) порождают именно такой язык [2]. Поэтому актуальна задача организации более выразительных запросов, используя контекстно-свободные грамматики.

В 2010 году был предложен алгоритм синтаксического анализа GLL [13], который основан на идее нисходящего анализа и поддерживает

любые КС-грамматики. В исследовательском проекте YaccConstructor [17] реализован данный алгоритм с возможностью проводить синтаксический анализ графов с КС-ограничениями. Однако графы поддерживаются с метками только на рёбрах.

1. Постановка задачи

Целью данной работы является добавление в проект YaccConstructor возможности выполнения запросов с контекстно-свободными ограничениями к графу с метками на вершинах и рёбрах. Для её достижения были поставлены следующие задачи:

- реализовать возможность поиска путей в графе с помеченными вершинами и рёбрами по заданной КС-грамматике;
- реализовать набор функций для получения и обработки результатов;
- провести апробацию для оценки производительности.

2. Обзор

2.1. Синтаксический анализ графов

Для поиска путей в графе существует множество инструментов, позволяющих находить пути по регулярным грамматикам. Решений для поиска путей по КС-грамматике не так много, в особенности для графов с метками на вершинах и рёбрах.

2.1.1. Subgraph Queries by Context-free Grammars

В работе [14] решалась задача извлечения связного подграфа, состоящего из путей между двумя исходными вершинами, из графа с метками на вершинах и рёбрах. Класс подходящих путей описывается с помощью контекстно-свободной грамматики. Для синтаксического анализа используется алгоритм Earley, работающий в худшем случае за время $O(n^3)$. Однако, поиск путей производится не в исходном графе с метками на вершинах и рёбрах, а в преобразованном. Перед началом работы алгоритма из исходного получают новый двудольный граф с метками только на рёбрах. В новом графе число вершин и рёбер увеличивается на первоначальное количество вершин. Даже при небольших входных данных и для путей длины не больше 8 алгоритм работает 240 секунд [14], что делает его мало применимым на практике.

2.1.2. Запросы к RDF-графам

Одним из распространённых способов представлять данные в удобном для обработки виде является модель RDF. Данные, записанные в RDF, представляют собой набор триплетов субъект–предикат–объект. В совокупности они образуют помеченный ориентированный граф. Многие данные в биоинформатике представлены именно в таком формате.

Самым популярным языком для запросов к данным, представленным в формате RDF, является язык SPARQL [10]. Однако, он позволяет описывать только расширенные регулярные выражения. В статье [2] авторы описали алгоритм для поиска путей в RDF-графе, при-

надлежащих КС-языку, а также предложили язык csSPARQL, поддерживающий КС-грамматики. Показано, что сложность алгоритма $O((|N| * |G|)^3)$, где N — нетерминалы входной грамматики, G — RDF-граф [2].

2.1.3. Conjunctive Context-Free Path Queries

Также задача выполнения КС-запросов к графу решалась в статье [5]. В данной работе был предложен язык CCFPQ для описания КС-запросов к графам и разработан алгоритм поиска путей, основанный на алгоритме синтаксического анализа CYK. Однако в статье запросы выполняются к графу без меток на вершинах. Для того, чтобы использовать этот алгоритм для графа с помеченными вершинами и рёбрами, необходимо сначала преобразовать граф, что потребует дополнительных ресурсов.

2.2. YaccConstructor

На кафедре Системного программирования в лаборатории языковых инструментов разрабатывается проект YaccConstructor. Это платформа для исследований в области синтаксического анализа, написанная на языке F#. YaccConstructor позволяет создавать синтаксические анализаторы и имеет модульную архитектуру.

В рамках YaccConstructor реализован алгоритм, который применяет GLL для синтаксического анализа графов с метками на рёбрах. Исходная грамматика описывается на языке спецификации грамматик YARD [16], а объект, в котором требуется найти пути, удовлетворяющие исходной КС-грамматике, должен реализовывать интерфейс IParserInput. В результате работы алгоритма получается SPPF [11]. Это структура данных, которая эффективно хранит все деревья разбора, получаемые при синтаксическом анализе.

2.3. QuickGraph

В лаборатории языковых инструментов также поддерживается библиотека QuickGraph для платформы .NET, которая содержит различные реализации графов и алгоритмы для них. Для исполнения запросов к графам разрабатывается расширение библиотеки QuickGraph, позволяющее получать результаты запросов, например, в виде подграфа или множества путей.

3. Реализация

3.1. Синтаксический анализ графов

Для синтаксического анализа графов с метками на вершинах и рёбрах с помощью алгоритма GLL в YaccConstructor необходимо реализовать интерфейс IParserInput (рис. 1), который содержит необходимые функции для работы алгоритма.

Во время исполнения алгоритм использует номера позиций во входном объекте, значит все позиции в объекте должны иметь уникальный номер. В случае графа с метками на вершинах и рёбрах, все вершины нумеруются чётными номерами, а все исходящие рёбра из вершины с позицией k имеют позицию k+1. Это позволит быстро определять, является текущая позиция вершиной (чётный номер) или ребром (нечётный номер), что потребуется для получения следующих терминалов в графе по текущей позиции. Нумерация позиций осуществляется при добавлении рёбер к графу.

Одна из функций интерфейса принимает в качестве параметров текущую позицию во входе и некоторую функцию, которая применяется к следующим позициям и следующим токенам. Это необходимо для того, чтобы проверять, возможен ли дальнейший разор строки в зависимости от текущей позиции в грамматике. В реализованном интерфейсе следующие позиции и терминалы в графе определяются в зависимости от чётности текущей позиции k. Если k чётная, то следующим токеном будет метка на вершине с соответствующей позицией, а следующей позицией будет $k + 1$, обозначающая все исходящие рёбра из вершины k. Если k нечётная, то следующими токенами будут метки на всех исходящих рёбрах из вершины $k - 1$, а следующими позициями будут номера вершин, в которые входят исходящие рёбра.

При получении следующих токенов и позиций в графе часто требуется получать все исходящие рёбра из указанной вершины. Поэтому для представления графа используется структура AdjacencyGraph из библиотеки QuickGraph, которая позволяет делать это эффективно.

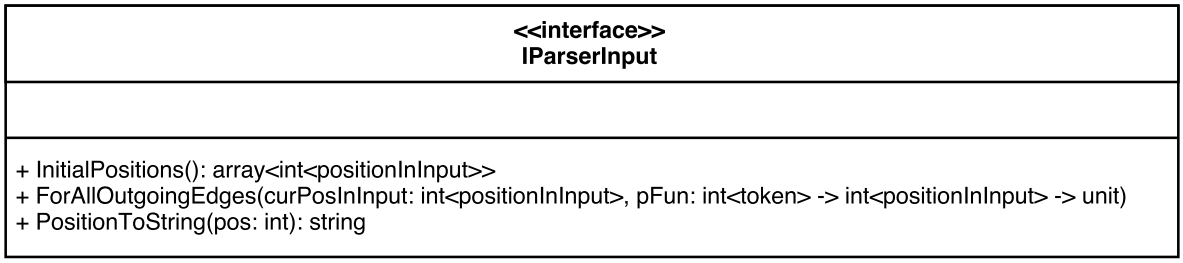

Рис. 1: Интерфейс IParserInput

Также для графа можно задать вершины, с которых будет начинать работу алгоритм и вершины, являющиеся конечными для синтаксического анализа. Для всех указанных вершин определяется их позиция в графе или, если вершина не найдена, вызывается сообщение об ошибке.

Для проверки работы алгоритма были написаны тесты, проверяющие корректность синтаксического анализа графов с различными КС-грамматиками.

3.2. Обработка и представление результатов запроса

Как уже было сказано, результатом работы алгоритма является структура данных SPPF, которая хранит все деревья разбора. В расширении библиотеки QuickGraph реализован набор функций, позволяющий извлекать из SPPF множество путей или подграф. Но подграф извлекается с метками только рёбрах, а вершинами являются позиции в исходном графе. Для получения графа с метками на вершинах и рёбрах из данного подграфа для каждой вершины с чётной позицией (ей соответствует вершина в исходном графе) извлекаются метки с трёх следующих рёбер, которые соответствуют метке на начальной вершине, метке на ребре и метке на конечной вершине. Данные три метки образуют ребро для нового графа. После добавления всех таких рёбер получается подграф с метками на вершинах и рёбрах.

Для наглядного представления результатов запроса, полученных в

виде подграфа, также реализована печать графа с метками на вершинах и рёбрах в dot-файл. Этот формат удобен для графического представления графов.

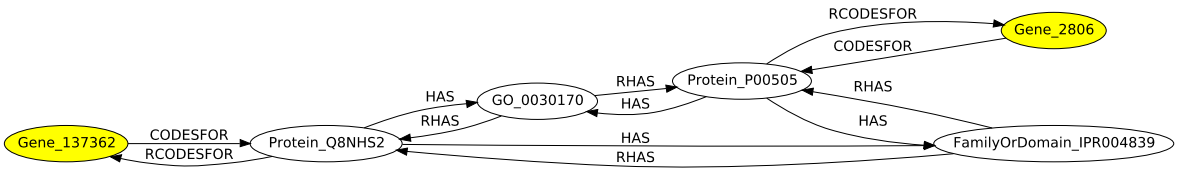

Рис. 2: Подграф, состоящий из путей между похожими генами

4. Экспериментальное исследование

В рамках данной работы были произведены эксперименты для реализованного алгоритма, который выполняет запросы к графу с метками на вершинах и рёбрах. Для этого были собраны данные и написан подходящий КС-запрос.

4.1. Данные

Существует большое количество биологических баз данных с открытым доступом, информация в которых может быть представлена как помеченный граф, в котором вершины соответствуют сущностям (протеины, гены, фенотипы), а рёбра отношениям между ними (взаимодействует, кодирует). Пути между вершинами позволяют найти новые связи в данных, либо показывают уже известные отношения. Подграф, построенный на всех найденных путях, более наглядно демонстрирует связи между вершинами.

Реальный набор биологических данных был собран из разных баз данных, находящихся в открытом доступе: Entrez Gene (информация о генах) [3], UniProt (протеины) [15], Gene Ontology (биологические процессы) [4], STRING (связи между протеинами) [12], InterPro (семейства белков) [7], KEGG (связи между генами) [8], HomoloGene (группы гомологий генов) [6]. Данные были ограничены набором из пяти организмов: Homo sapiens, Rattus norvegicus, Mus musculus, D. melanogaster и C. elegans. Объединенные в один файл данные состоят из троек: субъект, отношение, объект. Такие тройки образуют помеченный ориентированный граф.

```
[<Start>]
        s : gene
        v : protein | gene | GO | PATHWAY | FAMDOM
            | HOMOLOGENE
  similar : CODESFOR v RCODESFOR | BELONGS v RBELONGS
            | HAS v RHAS | HOMOLOGTO v RHOMOLOGTO
  protein : protein similar PROTEIN | PROTEIN
     gene : gene similar GENE | GENE
```

Рис. 3: Грамматика на языке YARD, задающая похожие гены

```
[<Start>]
        s : gene
        v : protein | gene | GO | PATHWAY | FAMDOM
            | HOMOLOGENE
  similar : CODESFOR v RCODESFOR | BELONGS v RBELONGS
            | HAS v RHAS | HOMOLOGTO v RHOMOLOGTO
       ps :  (PROTEIN similar) *[1..2]
  protein : ps PROTEIN | PROTEIN
       gs : (GENE similar) *[1..2]
     gene : gs GENE | GENE
```

Рис. 4: Итоговая грамматика с ограничениями, задающая похожие гены

4.2. Запросы

Все вершины в полученном графе имеют уникальную метку. Но для удобства будем различать их по типу: гены, фенотипы и т.д. Назовём две вершины в графе похожими, если они одного типа и имеют рёбра одного типа к похожим вершинам. Это определение рекурсивно. Таким образом, путь между похожими вершинами представляет собой палиндром, который нельзя задать с помощью регулярной грамматики.

На рисунке 3 показана KC-грамматика на языке YARD, задающая класс путей, в которых начальный и конечный терминалы являются похожими генами. Для поиска похожих генов нетерминал gene, обозначающий последовательность похожих генов, указывается стартовым.

Нетерминал similar задаёт отношение схожести генов и протеинов: существуют рёбра одного типа (например, BELONGS и RBELONGS) к похожим вершинам, которые обозначаются нетерминалом v.

В статье [14] при тестировании производительности длину путей ограничивали от 4 до 8. В алгоритме GLL в проекте YaccConstructor нет простого способа добавить такое ограничение. Поэтому, с целью уменьшения длины пути в грамматику были добавлены нетерминалы ps и gs, ограничивающие последовательность похожих протеинов и генов соответственно от одного до двух. На рисунке 4 показана итоговая грамматика с описанными выше ограничениями, задающая похожие гены.

На рисунке 2 показан подграф, который является результатом выполнения описанного выше запроса на небольших входных данных. Данный подграф содержит пути между похожими генами $Gene\_2806$ и $Gene\_137362$. Из рисунка видно, что эти гены кодируют похожие протеины, которые имеют рёбра одного типа has и $-has$ к одним и тем же вершинам $FamilyOrDomain\_IPR004839$ и $GO\_0030170$. То есть гены удовлетворяют описанному определению похожих вершин.

4.3. Производительность

Для оценки производительности была проведена серия экспериментов. Замеры произведены на графах с метками на рёбрах и вершинах, которые получены из описанных выше данных. Размер графов от 2 до 20 тысяч рёбер. В качестве запроса используется грамматика для поиска похожих генов. В статье [14] был проведён похожий эксперимент, но там длина пути ограничивалась от 4 до 8. В данной работе добиться такого ограничения не удалось. Для сокращения времени выполнения запроса используется грамматика с ограничением последовательности похожих генов и протеинов (рис. 4). На рисунке 5 показано время выполнения запроса с исходной грамматикой (рис. 3) и грамматикой с ограничениями. Запрос с ограничениями на графе с 20 тысячами рёбер выполняется примерно на 50 сек быстрее.

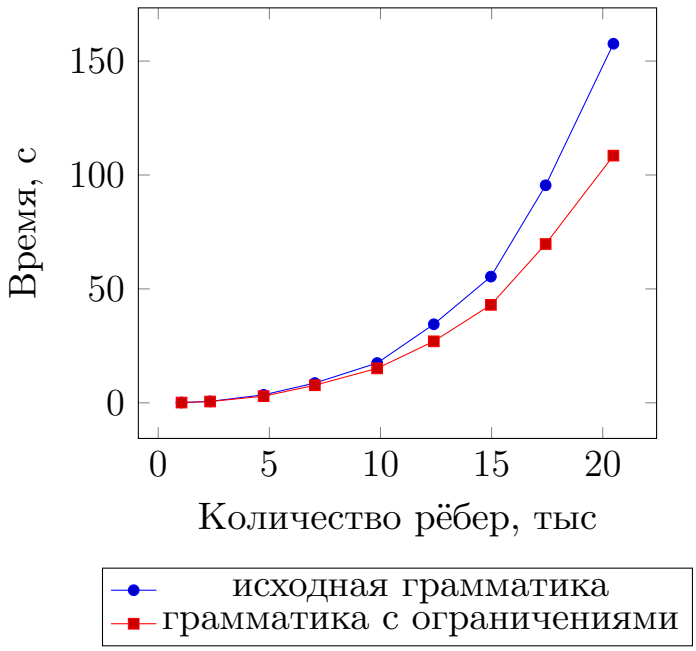

Рис. 5: Время выполнения исходного запроса и запроса с ограничениями.

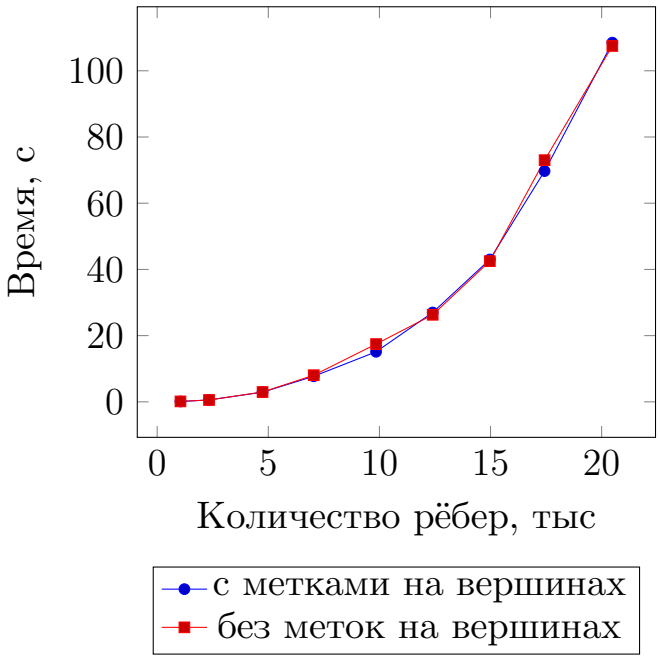

Рис. 6: Время выполнения запросов к исходному графу и к преобразованному.

Также была произведена серия экспериментов для сравнения производительности с алгоритмом GLL для графов без меток на вершинах. Замеры проведены на тех же данных, но с предварительным преобразованием графа в вершинно–рёберной граф (вершины с метками заменяются на рёбра). Результаты приведены на рисунке 6. Из графика видно, что выполнение запроса двумя способами занимает примерно одинаковое количество времени.

Таким образом, эксперименты показали, что реализованный алгоритм работает за приемлемое время на графах с числом рёбер до 20 тысяч. По сравнению с алгоритмом для графов с метками только на рёбрах выигрыша нет. Однако реализованный алгоритм избавляет от необходимости преобразовывать граф в вершинно–рёберный, что позволяет сэкономить ресурсы при больших входных данных.

5. Заключение

В ходе работы получены следующие результаты:

- реализована возможность поиска путей в графе с помеченными вершинами и рёбрами по заданной КС-грамматике;
- написаны тесты;
- реализован набор функций для получения и обработки результатов;
- проведены экспериментальные исследования.

5.1. Дальнейшее направление работ

Как показали результаты экспериментов, выполнение КС-запросов к графам уже с 20 тысячами рёбер занимает больше 100 секунд. Для уменьшения времени работы алгоритма и для выполнения запросов к графам большего размера исследовать возможность ограничения длины искомых путей в реализованном алгоритме GLL в YaccConstructor.

Список литературы

[1] Abiteboul Serge, Vianu Victor. Regular path queries with constraints // Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems / ACM. — 1997. — P. 122–133.

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

[3] Entrez Gene. — URL: `https://www.ncbi.nlm.nih.gov/gene`.

[4] Gene Ontology. — URL: `http://www.geneontology.org/page/download-ontology`.

[5] Hellings Jelle. Conjunctive context-free path queries. — 2014.

[6] HomoloGene. — URL: `https://www.ncbi.nlm.nih.gov/homologene`.

[7] InterPro. — URL: `https://www.ebi.ac.uk/interpro/download.html`.

[8] KEGG. — URL: `http://www.genome.jp/kegg/pathway.html`.

[9] Koschmieder André, Leser Ulf. Regular path queries on large graphs // Scientific and Statistical Database Management / Springer. — 2012. — P. 177–194.

[10] Prud'hommeaux Eric, Seaborne Andy. SPARQL Query Language for RDF. W3C Recommendation, January 2008. — 2008.

[11] Rekers Joan Gerard. Parser generation for interactive environments : Ph. D. thesis / Joan Gerard Rekers ; Universiteit van Amsterdam. — 1992.

[12] STRING. — URL: `https://string-db.org/`.

[13] Scott Elizabeth, Johnstone Adrian. GLL parsing // Electronic Notes in Theoretical Computer Science. — 2010. — Vol. 253, no. 7. — P. 177–189.

[14] Sevon Petteri, Eronen Lauri. Subgraph queries by context-free grammars // Journal of Integrative Bioinformatics (JIB). — 2008. — Vol. 5, no. 2. — P. 157–172.

[15] UniProt. — URL: `http://www.uniprot.org/uniprot/?query=reviewed:yes`.

[16] YaccConstructor. YARD // YaccConstructor official page. — URL: `http://yaccconstructor.github.io/YaccConstructor/yard.html`.

[17] YaccConstructor. YaccConstructor // YaccConstructor official page. — URL: `http://yaccconstructor.github.io`.