Д. Г. АРСЕНЬЕВ, Д. Е. БАСКАКОВ, В. П. ШКОДЫРЕВ
Санкт-Петербургский политехнический университет Петра Великого, Санкт-Петербург

ИНТЕЛЛЕКТУАЛЬНЫЕ ГРАФОВЫЕ МОДЕЛИ ОБРАБОТКИ СВЯЗАННЫХ ДАННЫХ

В связи с распространением различных интеллектуальных датчиков, устройств интернета вещей, смартфонов, автономных транспортных систем, различных систем промышленной и домашней автоматизации генерируется беспрецедентное количество данных, в том числе и интеллектуально-связанных между собой. Связанные данные позволяют выстраивать сложные и разнообразные отношения между объектами и субъектами реального мира. К сожалению, современные системы обработки больших данных и модели машинного обучения крайне плохо подходят для работы с такими динамически связанными данными, особенно в случае систем реального времени. Мы обсуждаем современные и перспективные подходы работы с такими данными с использованием графовых моделей анализа.

Введение. Давайте представим, что у нас есть сеть передачи данных, как на рис. 1. При этом в задачи администратора такой сети входит в том числе и диагностика неисправностей и сбоев, которые происходят в такой сети. Администратору часто приходится отвечать на следующие вопросы:

1. Каково было состояние сети в какое-то время?
2. Были ли какие-то каналы сети или башни перегружены и в какое время?
3. Каковы причины такого поведения сети и конкретных устройств?
4. Когда закончилась данная проблема и что происходило через час?
5. Что нужно сделать, дабы такая проблема не повторялась?

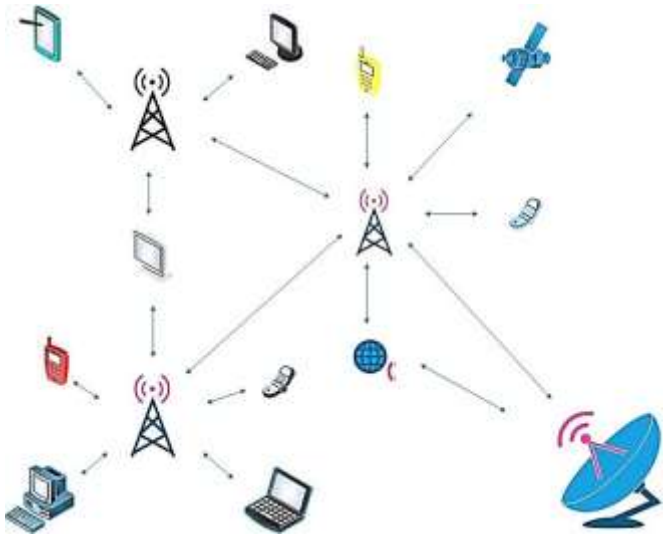

Рис. 1. Мобильная сеть передачи данных

Рассмотрим в качестве другого примера работу инвестиционного банка или биржи. Человек или отдел, которые отвечают на бирже за техническое состояние и обслуживание сетей и устройств передачи данных, должны уметь отвечать на ряд непростых вопросов, а именно:

1. От каких факторов зависит нагрузка на сеть передачи данных?
2. Как в течение дня данная нагрузка распределяется и какой она будет, например, завтра в 10.30?
3. По какой причине ряд клиентов не смогли сегодня с 13. 00 до 13. 14 вовремя выставить торговые заявки и что нужно сделать, дабы такое впредь не повторялось?

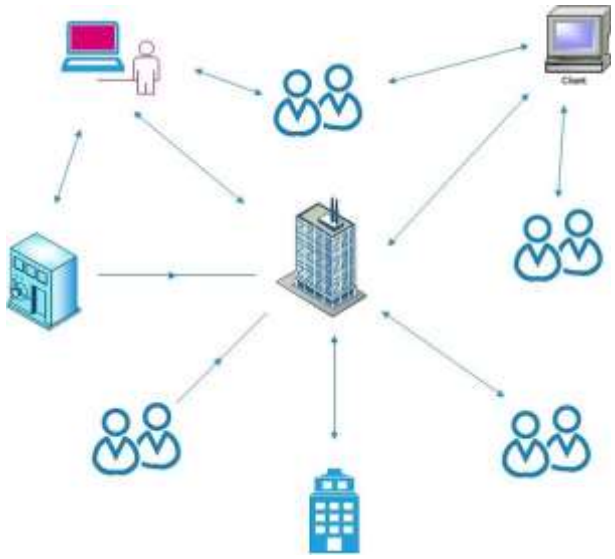

Рис. 2. Юридические и физические лица, работающие с банком

Мы можем также предположить, что для анализа сценариев поведения системы администраторы и технические специалисты мобильной сети используют продвинутые средства анализа, в том числе и пакеты машинного и глубокого обучения вида TensorFlow [1] или Scikit Learn [2]. Но у описанных инструментов выше есть две существенные проблемы. Во-первых, они не предназначены для работы с данными в режиме реального времени. Ну и во-вторых, данные инструменты крайне плохо подходят для работы с данными очень больших размеров. Налицо проблема динамичности и масштабируемости такой системы анализа, особенно в режиме реального времени.

Система на рис. 2 выглядит чуть иначе с точки зрения постановки целей и задач. Основная задача аналитика такой системы – выявление прежде всего финансовых махинаций или потенциально опасных операций или сделок, которые могут привести, например, к отрицательному балансу счета. В таких случаях аналитики используют модели машинного обучения и алгоритмы, которые позволяют в режиме почти реального времени выявлять потенциально опасные и проблемные транзакции, оценивать стороны таких операций, историю и назначение платежей. На основании такого анализа возникают некие новые модели и шаблоны потенциально опасных операций, которые и позволяют далее модернизировать алгоритм обучения. Инструменты могут быть примерно схожими. Помимо описанных ранее TensorFlow [1] и Scikit Learn [2] можно использовать инструмент анализа, управления, динамики и оценки структуры сложных сетей NetworkX [3]. Данный пакет содержит в последней версии 62 алгоритма работы с графами, несколько функций, а также более 26 способов генерации графов различных типов. При этом в случае с банком анализ операций может быть значительно более сложным в силу большого количества данных даже небольшого размера и сложности обнаружения таких операций в принципе, которые зачастую мошенниками проводятся под видом законных. В любом случае, проблема *динамичности* и *масштабируемости* имеет место быть и здесь. При этом данные, с которыми приходится работать аналитикам таких сетей, носят не какой-то обезличенный характер. Данные наделены весьма обширными семантическими причинно-следственными связями, то есть мы имеем дело со *связанными данными*.

Постановка задачи. На примере сложных технических систем мобильной связи и банковской системы расчетов стоит задача научиться анализировать в режиме реального времени события, которые формируют *связанные данные* и сложные семантические связи различного свойства. Как должна быть устроена такая система? Какие алгоритмы и подходы следует использовать при такой работе? В чем недостаток существующих систем? Об этом и постараемся рассказать ниже.

Методы решения.

Связанные данные. Под *связанными данными* мы понимаем данные, между которыми заданы или известны некоторые отношения в виде:

1. Предикатов (простейший случай).

2. Связей (один к одному, один ко многим, многие ко многим).
3. Путей (последовательность вершин, в которой каждая вершина соединена со следующим ребром) [4].
4. Графические вероятностные модели [5].
5. Неструктурированные отношения [6].

Области использования динамических связанных данных весьма обширны и конечно не ограничиваются только сферами сотовой связи или банкинга. Различные мобильные сенсоры и устройства интернета вещей, автономные транспортные средства [7], дальнейшее целостное развитие концепции связанных данных в парадигме «умные города» [8], – вот лишь неполный перечень возможных аспектов использования умных сенсоров для получения, передачи и обработки данных. При этом важно заметить, что существующие системы обработки данных для интернета вещей обладают несколькими существенными недостатками или ограничениями.

Хранение. Анализ и обработка большого объема связанных данных требуют совершенно новых подходов для построения высокопроизводительных систем, работающих в режиме реального времени. Данные генерируются каждый день, их извлечение и обработка требует нетривиальных подходов и концепций для организации эффективной работы.

Производительность. Система обработки связанных данных должна не только в режиме реального времени уметь принимать эти данные, но и уметь на основе полученных данных принимать те или иные решения, а также уметь объяснять и представлять, что же происходит на самом деле. На рынке есть немало решений, которые по-своему решают те или иные задачи, описанные выше. Такие решения, как NVIDIA Tegra [9], Altair Monarh [10], ASAP [11] заслуживают, безусловно, внимания при решении задач определенного типа.

Параллельная обработка связанных данных. Параллельные системы обработки данных стали крайне популярными в последние годы. При этом все большое распространение получает использование таких подходов при работе с большими и распределенными данными. В таких системах достаточно часто используются упрощенные абстракции данных и операторы, которые разработчики используют для достижения желаемых целей.

Одним из наиболее ярких представителей таких систем является MapReduce [12], который содержит только два оператора: *map* и *reduce.* MapReduce—модель распределенных вычислений, представленная компанией Google [13], используемых для параллельных вычислений над очень большими данными, вплоть до нескольких петабайт, наборами данных в компьютерных кластерах. Популярность MapReduce среди разработчиков и инженеров привела к появлению большого количества решений и систем, например *Apache Flink* [14]. При этом дальнейшее развитие проекта MapReduce привело к созданию систем нового поколения с более продвинутыми движками, сервисами и функциями, например *Naiad* [15] и *Spark* [16]. Данные системы содержат новые операторы, декларативные интерфейсы [17], более мощные алгоритмы управления кэшированием в память для снижения нагрузки при обработке данных и т.д.

Параллельная обработка графов. В этом разделе мы кратко рассмотрим параллельную обработку графов, касаясь их представления, абстракции и оптимизации.

Свойства графовых моделей. Под *графами* мы будем понимать абстрактные структуры данных, описывающих структурные отношения между связанными данными и объектами, которые являются носителями этих данных и свойств [18]. При этом свойства могут включать и метаданные (например, профили пользователей и отметки времени, количество соседей и т. д.). В качестве примера на рис. 1 такими метаданными обладают как базовые станции (содержат общий объем переданного трафика, количество активных пользователей, число сбоев и их возможные причины), так и пользователи (содержат данные о количестве звонков и сообщений). В таком случае администратор сети может создавать различные графовые модели сети в зависимости от времени, количества пользователей, отношения между ними и данными, которыми они обмениваются. Графы логически представлены и в MapReduce как пара наборов свойств вершин и ребер и их отношений с другими вершина посредством ребер. Это позволяет выстраивать сложные композиции в рамках потока данных и его анализа [19].

Альтернативным способом представления связанных данных является использование модели Resource Description Framework (RDF) [20]. Каждое свойство объекта в модели RDF описывается в виде триплета: *субъект – предикат (свойство) – объект:*

Рис. 3. Триплет RDF

На наш взгляд, использование графовых моделей представления знаний более соответствует анализу связанных данных, нежели модель RDF.

Граф – параллельная обработка данных. Современные системы обработки графовых данных дают мощные инструмент для их использования в том числе и при работе с параллельными данными. Типовая граф-параллельная абстракция состоит из графа и программы (в общем случае), которая работает в одной из вершин или в нескольких. Допускается использование до нескольких программ в одной вершине. Каждая программа запускается индивидуально и взаимодействует с соседними вершинами по какому-то алгоритму: передача общего состояния всем *соседним вершинам* (например, GraphLab [21]) или передача сообщений конкретным вершинам Pregel [22]. Дальнейшим развитием такого подхода является разложение Gather-Apply-Scatter (GAS) [23], в рамках которого *вершинная программ* использует три параллельные фазы: сбор информации и данных от смежных верши и ребер, применение функции к полученным данным и передача результатов работы функции посредство ребер другим вершинам.

```
def Gather(u, v) = Accum
def Apply(v, Accum) = vnew
def Scatter(v, j) = jnew, Accum
```

Листинг 1. Gather - Apply - Scatter (GAS) в реализации PowerGraph [24]

Тензорная обработка данных. Графовая обработка связанных данных имеет ограничение в виде одномерного пространства. При этом современные и перспективные системы глубокого обучения используют уже не векторные или матричные данные, а тензорные данные неограниченной размерности [1]. Дальнейшее развитие тензорной обработки данных сдерживается в том числе и отсутствием серьезных теоретических оснований для такой деятельности. Тем не менее, уже появилось понятие Adaptive Tensor Learning и Tensor Networks [25].

Стягивание ребра графа. В теории графов *стягивание ребра* – унарная операция, которая удаляет ребро из графа, а до этого связанные ребром вершины сливаются в одну. Есть и другая схожая операция, как *отождествление вершин,* но с более слабыми ограничениями. Назовем $\kappa -$ связанный граф G *минимальным по стягиванию, если для любого ребра* $e \in E(G)$ граф $G \cdot e$ не является $\kappa -$ связанным. Стягивание графа на примере алгоритма Каргера [26] выглядит следующим образом: на произвольном ребре $e = \{u, v\}$ происходит объединение вершин графа u и v дв одну uv. Если удаляется вершина v, то каждое ребро вида $\{v, x\}$ заменяется на ребро вида $\{u, x\}$. Петли графа удаляются и граф более не содержит петель.

procedure contract (G= (V, E)):

while $|V| > 2$

choose $e \in E$ uniformly at random

$G \leftarrow G / e$

return the only cunt in G

Листинг 2. Псевдокод алгоритма Каргера

Алгоритм Каргера представляет собой равновероятный выбор случайного имеющегося ребра с дальнейшим объединением вершин. В случае работы со связанными данными операция стягивания ребер графа носит фундаментальный характер, ибо позволяет существенно упростить работу при анализе графов большого размера.

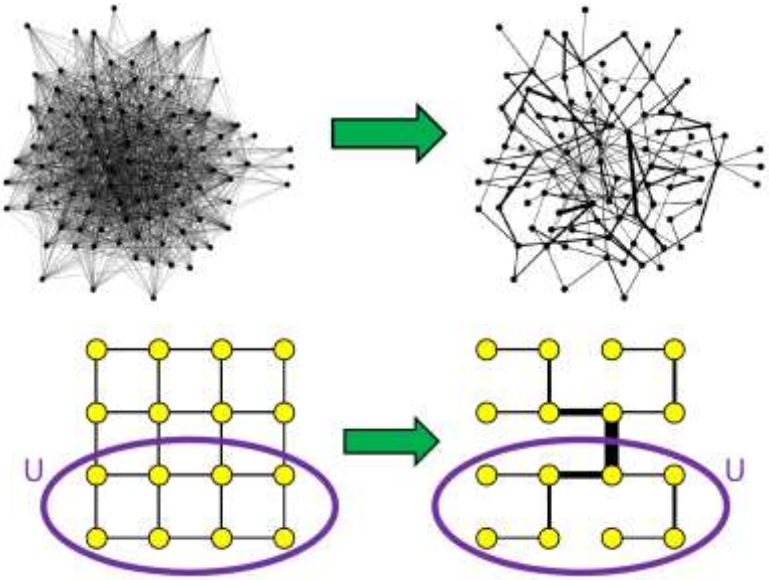

Рис. 4. Примеры стягиваний ребер графов

Анализ связанных данных. Наше видение анализа связанных данных с помощью графов состоит в следующем:

- Необходимо использовать случайным образом несколько алгоритмов для стягивания ребер графа.
- Необходимо использовать различные системы обработки графов с разными режимами работы: асинхронные или в режиме реального времени.
- Система должна создавать минимальную рабочую нагрузку на имеющиеся ресурсы.

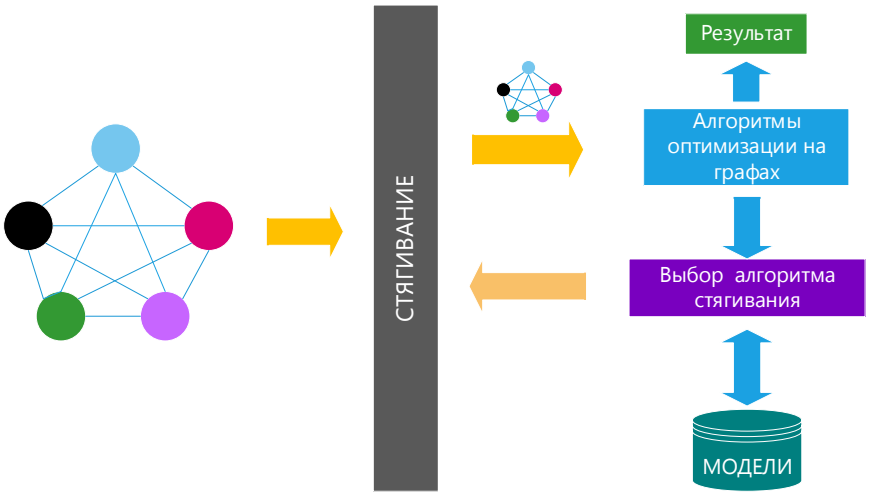

Рис. 5. Архитектура системы анализа связанных данных с использованием графов

Результаты. В рамках проведенного исследования была создана модель обработки связанных данных с использованием различных алгоритмов. Важно заметить, что за рамками данного изложения находится настройка параметров стягивания различных графов с использованием алгоритмов оптимизации. Дальнейшим развитием данной архитектуры видится использование методов машинного обучения для оптимизации параметров стягивания на графе. Также перспективным видится разработка теоретических основ многомерного стягивания в случае использования вместо обычных одномерных вершин тензоров n-размерности [1].

Выводы. Обработка связанных данных в режиме реального времени является, по нашему мнению, крайне перспективным направлением построения и анализа сложных иерархических структур. При этом такие структуры возникают часто весьма спонтанно как следствие отражения сложных взаимоотношений между объектами и субъектами в реальном мире. Актуальным

остается вопрос и дальнейшего развития как алгоритмов и методов такого анализа, так и построение теории поведения таких систем в целом.

Заключение. В работе мы показали наше видение развития анализа связанных данных с использованием графов. Архитектура предложенной системы требует дальнейшей практической реализации и доработки прежде всего с точки зрения автоматизации и применения методов машинного обучения. Именно в этом направлении и будут сосредоточены наши дальнейшие усилия, ибо практическая реализация таких систем как раз и демонстрирует эффективность предложенного подхода.

ЛИТЕРАТУРА

1. TensorFlow [Электронный ресурс]
2. Scikit Learn [Электронный ресурс] URL: https://scikit-learn.org/stable/
3. NetworkX [Электронный ресурс] URL: http://networkx.github.io/
4. Путь (теория графов) [Электронный ресурс] URL: https://ru.wikipedia.org/wiki/ Теория_графов
5. **Daphne Koller and Nir Friedman**. Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning. The MIT Press, 2009. 1268 pp.
6. **Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius** Zambaldi etc. Relational inductive biases, deep learning, and graph networks 2018. URL: https://arxiv.org/pdf1806.01261.pdf
7. Self-driving car [Электронный ресурс] URL: https://en.wikipedia.org/wiki/Self-driving_car
8. **Строев П.В.,** С.Б.Р. "Умный город "как новый этап городского развития, Грант РГНФ № 17-02-00269
9. NVIDIA Tegra [Электронный ресурс] URL: https://developer.nvidia.com/tegra-development
10. ALTAIR [Электронный ресурс] URL: https://www.altair.com/monarch/
11. ASAP [Электронный ресурс] URL: https://www.asapnetwork.org/
12. Hadoop.apache.org [Электронный ресурс] URL: https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html
13. Google [Электронный ресурс] URL: https://en.wikipedia.org/wiki/Google
14. Apache Flink [Электронный ресурс] URL: https://flink.apache.org/
15. **Derek G. Murray, Frank McSherry, Rebecca Isaacs, Michael Isard, Paul Barham, and Martin Abadi**. Naiad: A Timely Dataflow System // ACM. Farminton, Pennsylvania. 2013. Vol. Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles. SOSP ’13. pp. 439-455.
16. Spark.Apache.org [Электронный ресурс] URL: https://spark.apache.org/
17. **Кейно П.П., Силуянов А.В.** Разработка и внедрение интерпретатора декларативного языка моделирования Web-интерфейсов на высоконагруженных системах // Прикладная информатика, Vol. 10, No. 55, 201. pp. 55-70.
18. **Ian Robinson, Jim Webber, and Emil Eifrem**. Graph Databases. O’Reilly Media, Inc, 2013.
19. **Joseph Gonzalez, Reynold Xin, Ankur Dave, Daniel Crankshaw, Michael J. Franklin**. GraphX: Graph Processing in a Distributed Dataflow Framework // 11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14). Broomfield. Oct. 2014.
20. Resource Description Framework [Электронный ресурс] URL: https://ru.wikipedia.org/wiki/ Resource_Description_Framework
21. **Yucheng Low, Joseph Gonzalez, Aapo Kyrola, Danny Bickson, Carlos Guestrin, Joseph Hellerstein**. GraphLab: A New Framework For Parallel Machine Learning // https://arxiv.org/. 2010. URL: https://arxiv.org/ftp/arxiv/papers/1408/1408.2041.pdf
22. **Grzegorz Malewicz, Matthew H. Austern, Aart J.C Bik, James C. Dehnert, Ilan Horn, Naty Leiser, and Grzegorz Czajkowski**. Pregel: A System for Large-scale Graph Processing // Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data. Indianapolis, Indiana, USA. 2010. pp. 135-146.
23. **Joseph E. Gonzalez, Yucheng Low, Haijie Gu, Danny Bickson, and Carlos Guestrin**. PowerGraph: Distributed Graph-parallel Computation on Natural Graphs // Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation. OSDI’12. Hollywood, CA, USA. 2012. pp. 17-30.
24. github.com [Электронный ресурс] URL: https://github.com/jegonzal/PowerGraph
25. **Meraj Hashemizadeh, Michelle Liu, Jacob Miller, Guillaume Rabusseau.** Adaptive Tensor Learning with Tensor Networks // arxiv. 2020. URL: https://arxiv.org/pdf/2008.05437.pdf
26. URL: https://en.wikipedia.org/wiki/Karger%27s_algorithm

D.G. Arsenjev, D.E. Baskakov, V.P. Shkodyrev (Peter the Great St.Petersburg Polytechnic University, Saint Petersburg).

Requirements to Formatting the Papers to be Published in the Proceedings of the Multiconference on Control Problems

In connection with the proliferation of various intelligent sensors, IoT devices, smartphones, autonomous transport systems, various industrial and home automation systems, an unprecedented amount of data is generated, including intelligently linked to each other. Linked data allows you to build complex and varied relationships between objects and subjects in the real world. Unfortunately, modern big data processing systems and machine learning models are extremely poorly suited for working with such dynamically linked data, especially in the case of real-time systems. We discuss current and future approaches to working with such data using graph analysis models.