INTELLECTUAL GRAPH MODELS FOR RELATED DATA PROCESSING

D Arseniev¹, D. Baskakov² and V. Shkodyrev³

¹ Doctor of Technical Sciences Peter the Great St. Petersburg Polytechnic University, St. Petersburg, Russia
² Research Officer, Peter the Great St. Petersburg Polytechnic University, St. Petersburg, Russia
³ Doctor of Technical Sciences, Peter the Great St. Petersburg Polytechnic University, St. Petersburg, Russia

E-mail: vicerector.int@spbstu.ru, dmitry.e.baskakov@gmail.com, shkodyrev@mail.ru

Abstract. In connection with the wide spreading of various intelligent sensors, IoT devices, smartphones, autonomous transport systems, various industrial and home automation systems, an unprecedented amount of data is generated, including those intelligently linked to each other. Linked data allows you to build complex and varied relationships between objects and subjects of 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-proof approaches to working with such data using graph analysis models.

Introduction

Imagine that we have a data transmission network as in Figure 1. At the same time, the tasks of the administrator of such a network include, among other things, diagnostics of malfunctions and failures that occur in such a network. The administrator often has to answer the following questions:

1. What was the state of the network at some time?
2. Were any network channels or towers congested and at what time?
3. What are the reasons for this behavior of the network and particular devices?
4. When did the problem end and what happened an hour later?
5. What needs to be done so that this problem does not recur?
Consider the operation of an investment bank or exchange as another example. The person or department responsible for the technical condition and maintenance of networks and data transmission devices at the exchange must be able to answer a number of difficult questions, namely:

1. Which factors determine the load on the data transmission network?
2. How is this load distributed during the day and what will it be, for example, tomorrow at 10.30 a.m.?
3. For what reason a number of clients were unable today from 13:00 to 13.14 to place trade orders on time and what needs to be done so that this does not happen again?

We can also assume that administrators and mobile network technicians use advanced analysis tools to analyze system behavior scenarios, including machine and deep learning packages such as TensorFlow [1] or Scikit Learn [2]. However, the tools described above have two significant problems. First, they are not designed to work with real-time data. And second, these tools are extremely poorly suited for working with very large data. There is an obvious problem of the dynamism and scalability of such an analysis system, especially in real time.
The system in Figure 2 looks a little different in terms of setting goals and objectives. The main task of an analyst of such a system is to identify, first of all, financial fraud or potentially dangerous operations or transactions that can lead, for example, to a negative account balance. In such cases, analysts use machine learning models and algorithms that make it possible, in almost real time, to identify potentially dangerous and problematic transactions, evaluate the parties to such transactions, the history and purpose of payments. Based on this analysis, some new models and patterns of potentially dangerous operations appear, which allow further modernization of the learning algorithm. The tools can be roughly similar. In addition to the previously described Tensor-Flow [1] and Scikit Learn [2], you can use the tool for analysis, control, dynamics and assessment of the structure of complex networks NetworkX [3]. The latest version of this package contains 62 algorithms for working with graphs, several functions, as well as more than 26 ways to generate graphs of various types. At the same time, in the case of a bank, the analysis of transactions can be much more complicated due to the large amount of data, even of a small size, and the overall difficulty of detecting such transactions, which are often carried out by fraudsters under the guise of legitimate ones. In any case, the problem of dynamism and scalability also exists here. At the same time, the data that analysts of such networks have to work with are not impersonal in nature. The data are endowed with very extensive semantic cause-and-effect relationships, which means that, we are dealing with related data.

1. Formulation of the problem
Using the example of complex technical systems of mobile communications and a banking system of settlements, the task is to learn how to analyze in real time the events that form related data and complex semantic connections of various properties. How should such a system be arranged? What algorithms and approaches should be used for such work? What is the disadvantage of existing systems? We will try to talk about this below.

2. Solution methods
Linked data. By linked data we mean the data between which certain relationships are set or known in following forms:
1. Predicates (the simplest case).
2. Relationships (one to one, one to many, many to many).
3. Paths (a sequence of vertices with each vertex connected to the next edge) [4].
4. Graphical probabilistic models [5].
5. Unstructured relationships [6].

There are a lot of applications of dynamic linked idea, and of course they are not limited to cellular communications or banking. Various mobile sensors and devices of the Internet of Things, autonomous vehicles [7], further holistic development of the concept of connected data in the paradigm of "smart cities" [8] - this is just an incomplete list of possible aspects of using smart sensors for receiving, transmitting and processing data. It is important to note, however, that existing data processing systems for the Internet of Things have several significant drawbacks or limitations.

2.1. Storage.
Analyzing and processing large volumes of related data requires completely new approaches to building high-performance systems operating in real time. Data are generated every day; their extraction and processing require non-trivial approaches and concepts to organize effective work.

2.2. Performance
Besides receiving the data in real time, the system for processing related data should also be able to make certain decisions based on the data received, and to explain and realize what is really happening. There are a lot of solutions on the market that, in their own way, solve certain problems described above.
Such solutions as NVIDIA Tegra [9], Altair Monarh [10], ASAP [11] certainly deserve attention when solving problems of a certain type.

3. Parallel processing of related data

Parallel data processing systems have become extremely popular in recent years. At the same time, the use of such approaches is becoming increasingly widespread when working with large and distributed data. Such systems often use simplified data abstractions and operators used by the developers to achieve their desired goals.

One of the most prominent representatives of such systems is MapReduce [12] which contains only two operators: map and reduce. MapReduce is a distributed computing model presented by Google [13], used for parallel computing over very large data up to several petabytes, datasets in computer clusters. The popularity of MapReduce among developers and engineers has led to the emergence of a large number of solutions and systems, such as Apache Flink [14]. At the same time further development of the MapReduce project led to the creation of new generation systems with more advanced engines, services and functions, for example, Naiad [15] and Spark [16]. These systems contain new operators, declarative interfaces [17], more powerful algorithms for managing caching in memory to reduce the load during data processing, etc.

3.1. Parallel processing of graphs

In this section, we will take a quick look at parallel processing of graphs, touching on their representation, abstraction, and optimization.

3.1.1. Graph model properties. By graphs, we mean abstract data structures which describe structural relationships between related data and objects being the carriers of these data and properties [18]. However, the properties can also include metadata (for example, user profiles and timestamps, number of neighbors, etc.). As an example, in Figure 1, such metadata are possessed by both base stations (containing the total amount of transmitted traffic, the number of active users, the number of failures and their possible causes) and users (containing data on the number of calls and messages). In this case, the network administrator can create various graph models of the network depending on the time, the number of users, the relationship between them and the data they exchange. Graphs are logically represented in MapReduce as a pair of sets of properties of vertices and edges and their relationship with other vertices by means of edges. This allows one to build complex compositions within the data flow and its analysis [19]. An alternative way to represent related data is to use the Resource Description Framework (RDF) model [20]. Each property of an object in the RDF model is described as a triplet: subject - predicate (property) - object:

Subject Property Object

![Fig. 3. RDF triplet.](image)

In our opinion, the use of graph models of knowledge representation is more consistent with the analysis of related data than the RDF model.

3.1.2. Graph - parallel data processing. Modern systems for processing graph data provide a powerful tool for their use, including when working with parallel data. A typical graph-parallel abstraction consists of a graph and a program (in the general case) that works at one of the vertices or at several ones. It is allowed to use a few programs in one vertex. Each program runs individually and interacts with neighboring vertices according to some algorithm: transmitting the general state to all neighboring vertices (for example, GraphLab [21]) or sending messages to specific Pregel vertices [22]. Further
development of this approach is the Gather-Apply-Scatter (GAS) decomposition [23], within which the vertex program uses three parallel phases: collecting information and data from adjacent vertices and edges, applying a function to the received data, and transmitting the results of the function work by means of edges to other vertices.

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

Listing 1. Gather-Apply-Scatter (GAS) in PowerGraph implementation [24].

3.2. Tensor data processing

Graph processing of related data is limited to one-dimensional space. Moreover, modern and promising deep learning systems already use tensor data of unlimited dimension rather than vector or matrix data. [1]. Further development of tensor data processing is hindered, among other things, by the lack of serious theoretical grounds for such activities. However, the concept of Adaptive Tensor Learning and Tensor Networks has already appeared [25].

3.3. Contraction of an edge of a graph

In the graph theory, contraction of an edge is a unary operation which removes an edge from a graph, and before that, the vertices connected by this edge merge into one. There is another similar operation, known as vertex identification, but with weaker restrictions. We say that a $k$-connected graph $G$ is contraction-minimal if, for any edge $e \in E(G)$, the graph $G \cdot e$ is not $k$-connected. Graph contraction by the example of Karger's algorithm looks as follows: on an arbitrary edge $e = \{u, v\}$, the vertices of the graph $u$ and $v$ are combined into one $uv$. If the vertex $v$ is removed, then each edge of the form $\{v, x\}$ is replaced by an edge of the form $\{u, x\}$. [26]. The graph loops are removed and the graph no longer contains loops.

```python
procedure contract (G= (V, E)):
    while $|V| > 2$
        choose $e \in E$ uniformly at random
        $G \leftarrow G / e$
    return the only cut in $G$
```

Listing 2. Pseudocode for Karger's algorithm.

Karger's algorithm is an equiprobable selection of a random available edge with further union of the vertices. In the case of working with related data, the operation of contraction of the edges of the graph is of a fundamental nature, because it allows to significantly simplify the work when analyzing large graphs.
4. Related data analysis

Our vision for analyzing the related data using graphs is as follows:
1. It is necessary to randomly use several algorithms to contract the edges of the graph.
2. It is necessary to use different graphs processing systems with different modes of operation: asynchronous or in real time.
3. The system should have a minimum workload on the available resources.

5. Results

As part of the study, a model was created for processing related data using various algorithms. It is important to note that it is beyond the scope of this presentation to adjust the contraction parameters of
various graphs using optimization algorithms. Further development of this architecture sees the use of machine learning methods to optimize the contraction parameters on the graph. It also seems promising to develop the theoretical foundations of multidimensional contraction in the case of using n-dimensional tensors instead of the usual one-dimensional vertices [1].

6. Conclusions
The processing of related data in real time is, in our opinion, an extremely promising direction for the construction and analysis of complex hierarchical structures. Moreover, such structures often arise very spontaneously as a consequence of the reflection of complex relationships between objects and subjects in the real world. The question of the further development of both algorithms and methods of such analysis, and the construction of a theory of the behavior of such systems in general, remains relevant. Conclusion. In this work, we showed our vision of the development of analysis of related data using graphs. The architecture of the proposed system requires further practical implementation and refinement, primarily from the point of view of automation and application of machine learning methods. It is in this direction that our further efforts will be focused, because the practical implementation of such systems just demonstrates the effectiveness of the proposed approach.

References

[1] "TensorFlow," [Online].
[2] "Scikit Learn," [Online]. Available: https://scikit-learn.org/stable/.
[3] "NetworkX," [Online]. Available: http://networkx.github.io/.
[4] [Online]. Available: 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, p. 1268.
[6] Peter W. Battaglia, Jessica B. Hamrick, Victor Basil, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi etc, "Relational inductive biases, deep learning, and graph networks," 2018. [Online]. Available: https://arxiv.org/pdf/1806.01261.pdf.
[7] "Self-driving car," [Online]. Available: https://en.wikipedia.org/wiki/Self-driving_car.
[8] S. B. R. P.V. Stroyev, "Smart city as a new stage of urban development".
[9] "NVIDIA Tegra," [Online]. Available: https://developer.nvidia.com/tegra-development.
[10] "ALTAIR," [Online]. Available: https://www.altair.com/monarch/.
[11] "ASAP," [Online]. Available: https://www.asapnetwork.org/.
[12] "Hadoop.apache.org," [Online]. Available: https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html.
[13] "Google," [Online]. Available: https://en.wikipedia.org/wiki/Google.
[14] "Apache Flink," [Online]. Available: https://link.apache.org/.
[15] Derek G. Murray, Frank McSherry, Rebecca Isacs, Michael Isard, Paul Barham, and Martin Abadi, "Naïad: A Timely Dataflow System," in ACM, Farmington, Pennsylvania, 2013.
[16] "Spark.apache.org," [Online]. Available: https://spark.apache.org/.
[17] P.P. Keino, A.V. Siluyanov, "Development and implementation of an interpreter for a declarative language for modeling Web interfaces on high-load systems," Applied Informatics, vol. 10, no. 55, pp. 55-70, 201.
[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," in 11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14), Broomfield, Oct. 2014.
[20] "Resource Description Framework," [Online]. Available: 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," 2010. [Online]. Available: 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," in Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data, Indianapolis, Indiana, USA, 2010.

[23] Joseph E. Gonzalez, Yucheng Low, Haijie Gu, Danny Bickson, and Carlos Guestrin, "PowerGraph: Distributed Graph-parallel Computation on Natural Graphs," in Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation, OSDI'12. Hollywood, CA, USA, 2012.

[24] "github.com," [Online]. Available: https://github.com/jegonzal/PowerGraph.

[25] Meraj Hashemizadeh, Michelle Liu, Jacob Miller, Guillaume Rabusseau, "Adaptive Tensor Learning with Tensor Networks," 2020. [Online]. Available: https://arxiv.org/pdf/2008.05437.pdf.

[26] [Online]. Available: https://en.wikipedia.org/wiki/Karger%27s_algorithm.