neo4j

The #1 Platform for Connected Data

WHITE PAPER

Optimized Graph Algorithms in Neo4j

Use the Power of Connections to Drive Discovery

"...The tools of graph theory can be utilized in order to analyze the networks and obtain a better understanding of their overall construction. This approach has led to several groundbreaking discoveries on the nature of networks, crossing fields of research from biology, to social science and technology."

Albert-László Barabási
Director, Center for Complex Network Research
Northeastern University
Author of numerous network science books

White Paper

TABLE OF CONTENTS

Optimized Graph Algorithms in Neo4j

Use the Power of Connections to Drive Discoveries

Amy Hodler

Algorithms: The Graph Analysis Powerhouse

Graph algorithms are the powerhouse behind the analysis of real-world networks—from identifying fraud rings and optimizing the location of public services to evaluating the strength of a group and predicting the spread of disease or ideas.

Based on the unique mathematics of graph theory, these algorithms use the connections between data to evaluate and infer the organization and dynamics of complex systems. Data scientists use these penetrating graph algorithms to bring to light valuable information hidden in our connected data. They then use this analysis to iterate prototypes and test hypotheses.

A Practical Approach to Graph Analytics

Graph analytics have value only if you have the skills to use them and if they can quickly provide the insights you need. Therefore, the best graph algorithms are easy to use, fast to execute and produce powerful results.

For transactions and operational decisions, you need real-time graph analysis to provide a local view of relationships between specific data points. To discover the overall nature of networks and model the behavior of intricate systems, you need global graph algorithms that provide a broad view of patterns and structures across all data and relationships.

Other analytics tools layer graph functionality atop databases with non-native graph storage and computation engines. These hybrid solutions seldom support ACID transactions, which can ruin data integrity. Also, they must execute complicated JOINs for each query, crippling performance and wasting system resources.

Alternatively, you could maintain multiple environments for graph analytics, but then your algorithms aren't integrated with, nor optimized for a graph data model. This bulky approach is less efficient, less productive, more costly and greatly increases the risk of errors.

Optimized Graph Algorithms in Neo4j

To understand data connections, you need global graph algorithms that provide a broad view of patterns and structures across all data and relationships.

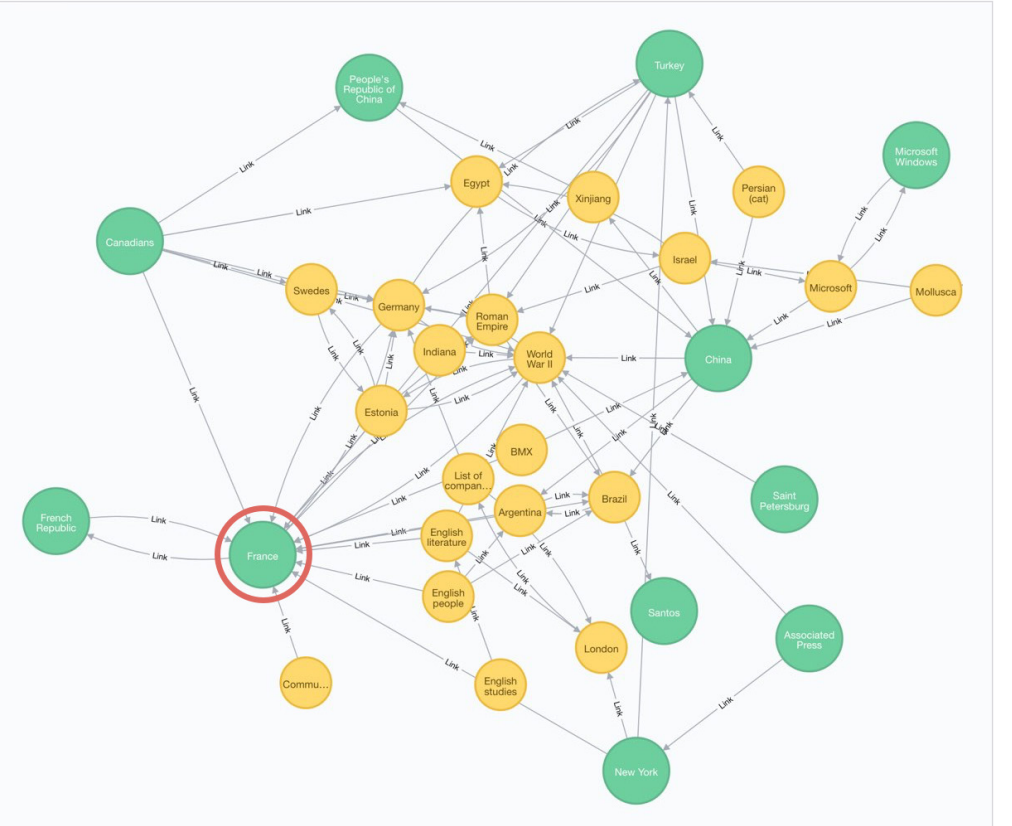
Real-time graph algorithms require exceptionally fast (millisecond-scale) results whereas global graph algorithms can be very computationally demanding. Graph analytics must have algorithms optimized for these different requirements with the ability to efficiently scale—analyzing billions of relationships without the need for super-sized or burdensome equipment. This kind of versatile scale necessitates very efficient storage and computational models as well as the use of state-of-the-art algorithms that avoid stalling or recursive processes.

Finally, a collection of graph algorithms must be vetted so your discoveries will be trustworthy, and include ongoing educational material so your teams will be up-to-date. With these fundamental elements in place, you can confidently make progress on your breakthrough applications.

Example: Analyzing Category Influence in Wikipedia

Let's look at an example of how to use Neo4j Analytics to analyze the most influential categories in Wikipedia searches. The graph below shows only the largest of 2.6 million clusters found with the most influential categories in green. It reveals that *France* has significant influence as a large cluster-category with many, high-quality transitive links.

The Neo4j Label Propagation algorithm grouped related pages as a cluster-category in 24 seconds and then PageRank was used to identify the most influential categories by looking at the number and quality of transitive links in 23 seconds (using 144 CPU machine and 32GB RAM of 1TB total, SSD).

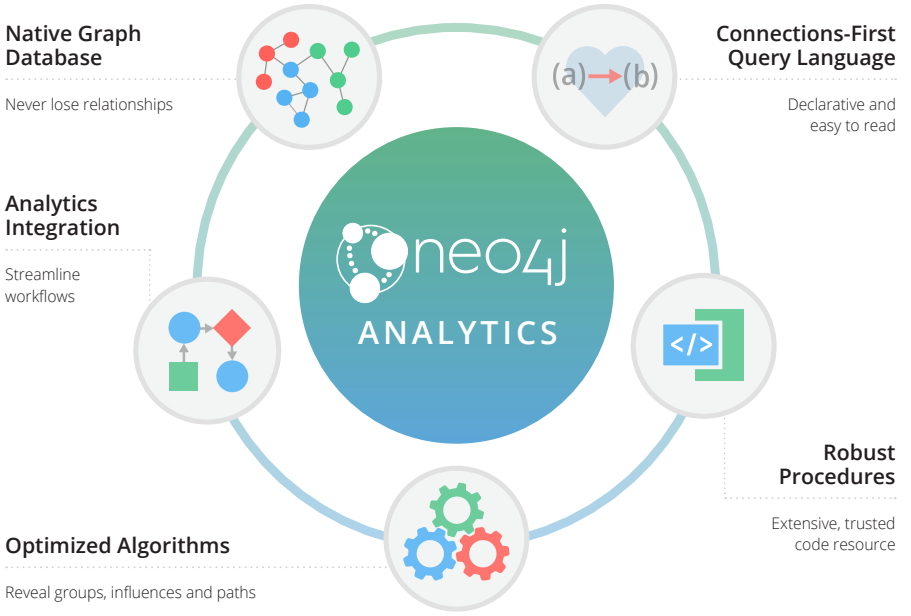


Representation of category influence on Wikipedia, using DBpedia's extracted links with 116 million relationships and 11 million nodes.

Optimized Graph Algorithms in Neo4j

The Neo4j Graph Analytics Platform

Neo4j offers a reliable and performant native-graph platform that reveals the value and maintains the integrity of connected data. First, we delivered the Neo4j graph database, originally used in online transaction processing with exceptionally fast transversals. Then we Exadded advanced, yet practical, graph analytic tools for data scientists and solutions teams.



Neo4j offers a reliable and performant native-graph platform that includes practical, graph analytics tools for data scientists and solutions teams.

Streamline Your Discoveries

We offer a growing, open library of high-performance algorithms for Neo4j that are easy to use and optimized for fast results. These algorithms reveal the hidden patterns and structures in your connected data around community detection, centrality and pathways with a core set of tested (at scale) and supported algorithms. The highly extensible nature of Neo4j enabled the creation of this graph library and exposure as procedures—without making any modification to the Neo4j database.

Model and predict complicated dynamics such as resource and information flows, propagation pathways and group resiliency.

These algorithms can be called upon as procedures (from our APOC library) and they're also customizable through a common graph API. This set of advanced, global graph algorithms is simple to apply to existing Neo4j instances so your data scientists, solutions developers and operational teams can all use the same native graph platform.

Neo4j also includes graph projection, an extremely handy feature that places a logical sub-graph into a graph algorithm when your original graph has the wrong shape or granularity for that specific algorithm. For example, if you're looking to understand the relationship between drug results for men versus women but your graph is not partitioned for this, you'll be able to temporarily project a sub-graph to quickly run your algorithm upon and move on to the next step.

Optimized Graph Algorithms in Neo4j

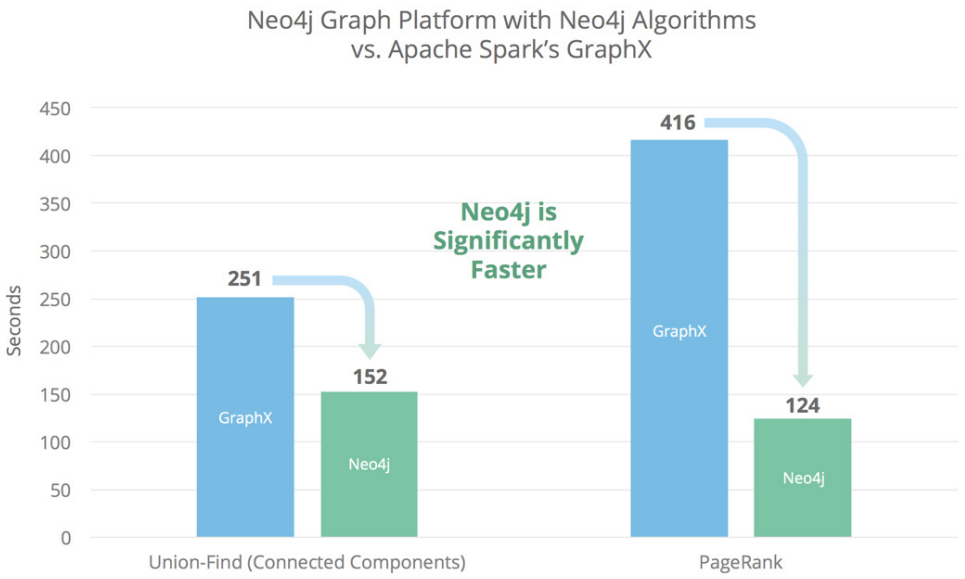
"Graphs are one of the unifying themes of computer science—an abstract representation that describes the organization of transportation systems, human interactions, and telecommunication networks. That so many different structures can be modeled using a single formalism is a source of great power to the educated programmer."

- Steven S. Skiena,
The Algorithm Design Manual

Example: High Performance of Neo4j Graph Algorithms

Neo4j graph algorithms are extremely efficient so you can analyze billions of relationships using common equipment and get your results in seconds to minutes, and in a few hours for the most complicated queries.

The chart below shows how Neo4j's optimized algorithms yields results up to three times faster than Apache Spark™ GraphX for Union-Find (Connected Components) and PageRank on the Twitter-2010 dataset with 1.4 billion relationships.



Twitter 2010 Dataset
1.47 Billion relationships with 41.65 million nodes

Spark GraphX Configuration[1]
Amazon EC2 cluster, 64-bit Linux, 128 CPUs, 68GB RAM, 2 drives

Neo4j Configuration
Server running 64-bit Linux, 128 CPUs, 55GB RAM, SSDs

Even more impressive, running the Neo4j PageRank algorithm on a significantly larger dataset with **18 billion relationships** and 3 billion nodes delivered results in only 1 hour and 45 minutes (using 144 CPUs and 1TB of RAM).

In addition to optimizing the algorithms themselves, we've parallelized key areas such as loading and preparing data as well as algorithms like Breadth-First Search and Depth-First Search where applicable.

The Power of Optimized Algorithms in Neo4j

Using Neo4j graph algorithms, you'll have the means to understand, model and predict complicated dynamics such as the flow of resources or information, the pathways that contagions or network failures spread, and the influences on and resiliency of groups. And because Neo4j brings together analytics and transaction operations in a native graph platform, you'll not only uncover the inner nature of real-world systems for new discoveries, but also develop and deploy graph-based solutions faster and have easy-to-use, streamlined workflows. That's the power of an optimized approach.

(1) Spark GraphX test results from *www.frankmcsherry.org/graph/scalability/cost/2015/01/15/COST.html*

Optimized Graph Algorithms in Neo4j

Find the shortest path or evaluate the availability and quality of routes.

Pathfinding and Traversal Algorithms

Algorithm Type	What It Does	Example Uses
Parallel Breadth-First Search (BFS)	Traverses a tree data structure by fanning out to explore the nearest neighbors and then their sub-level neighbors. It's used to locate connections and is a precursor to many other algorithms. BFS is preferred when the tree is less balanced or the target is closer to the starting point. It can also be used to find the shortest path between nodes or avoid recursive processes of DFS.	BFS can be used to locate neighbor nodes in peer-to-peer networks like BitTorrent, GPS systems to pinpoint nearby locations and social network services to find people within a specific distance.
Parallel Depth-First Search (DFS)	Traverses a tree data structure by exploring as far as possible down each branch before backtracking. It's used on deeply hierarchical data and is a precursor to many other algorithms. DFS is preferred when the tree is more balanced or the target is closer to an endpoint.	DFS is often used in gaming simulations where each choice or action leads to another, expanding into a tree-shaped graph of possibilities. It will traverse the choice tree until it discovers an optimal solution path (e.g., win).
Single-Source Shortest Path	Calculates a path between a node and all other nodes whose summed value (weight of relationships such as cost, distance, time or capacity) to all other nodes are minimal.	Single-Source Shortest Path is often applied to automatically obtain directions between physical locations, such as driving directions via Google Maps. It's also essential in logical routing such as telephone call routing (least cost routing).
All-Pairs Shortest Path	Calculates a shortest path forest (group) containing all shortest paths between the nodes in the graph. Commonly used for understanding alternate routing when the shortest route is blocked or becomes suboptimal.	All-Pairs Shortest Path can be used to evaluate alternate routes for situations such as a freeway backup or network capacity. It's also key in logical routing to offer multiple paths, for example, call routing alternatives.
Minimum Weight Spanning Tree (MWST)	Calculates the paths along a connected tree structure with the smallest value (weight of the relationship such as cost, time or capacity) associated with visiting all nodes in the tree. It's also employed to approximate some NP-hard problems such as the traveling salesman problem and randomized or iterative rounding.	MWST is widely used for network designs: least cost logical or physical routing such as laying cable, fastest garbage collection routes, capacity for water systems, efficient circuit designs and much more. It also has real-time applications with rolling optimizations such as processes in a chemical refinery or driving route corrections.

Optimized Graph Algorithms in Neo4j

Determine the importance of distinct nodes in a network of connected data.

Centrality Algorithms

Algorithm Type	What It Does	Example Uses
PageRank	Estimates a current node's importance from its linked neighbors and then again from their neighbors. A node's rank is derived from the number and quality of its transitive links to estimate influence. Although popularized by Google, it's widely recognized as a way of detecting influential nodes in any network.	PageRank is used in quite a few ways to estimate importance and influence. It's used to suggest Twitter accounts to follow and for general sentiment analysis. PageRank is also used in machine learning to identify the most influential features for extraction. In biology, it's been used to identify which species extinctions within a food web would lead to biggest chain-reaction of species death.
Degree Centrality	Measures the number of relationships a node (or an entire graph) has. It's broken into indegree (flowing in) and outdegree (flowing out) where relationships are directed.	Degree Centrality looks at immediate connectedness for uses such as evaluating the near-term risk of a person catching a virus or hearing information. In social studies, indegree of friendship can be used to estimate popularity and outdegree as gregariousness.
Closeness Centrality	Measures how central a node is to all its neighbors within its cluster. Nodes with the shortest paths to all other nodes are assumed to be able to reach the entire group the fastest.	Closeness centrality is applicable in a number of resources, communication and behavioral analysis, especially when interaction speed is significant. It has been used to identifying the best location of new public services for maximum accessibility. In social analysis, it can be used to find people with the ideal social network location for faster dissemination of information.
Betweenness Centrality	Measures the number of shortest paths (first found with BFS) that pass through a node. Nodes that most frequently lie on shortest paths have higher betweenness centrality scores and are the bridges between different clusters. It is often associated with the control over the flow of resources and information.	Betweenness Centrality applies to a wide range of problems in network science and can be used to pinpoint bottlenecks or likely attack targets in communication and transportation networks. In genomics, it has been used to understand the control certain genes have in protein networks for improvements such as better drug-disease targeting. Betweenness Centrality has also be used to evaluate information flows between multiplayer online gamers and expertise sharing communities of physicians.

Evaluate how a group is clustered or partitioned, as well as its tendancy to strengthen or break apart.

Community Detection Algorithms

Also known as clustering and partitioning

Algorithm Type	What It Does	Example Uses
Label Propagation	Spreads labels based on neighborhood majorities as a means of inferring clusters. This extremely fast graph partitioning requires little prior information and is widely used in large-scale networks for community detection. It's a key method for understanding the organization of a graph and is often a primary step in other analysis.	Label Propagation has diverse applications from understanding consensus formation in social communities to identifying sets of proteins that are involved together in a process (functional modules) for biochemical networks. It's also used in semi- and unsupervised machine learning as an initial preprocessing step.
Strongly Connected	Locates groups of nodes where each node is reachable from every other node in the same group following the direction of relationships. It's often applied from a depth-first search.	Strongly Connected is often used to enable running other algorithms independently on an identified cluster. As a preprocessing step for directed graphs, it can help quickly identify disconnected groups. In retail recommendations, it can help identify groups with strong affinities that then can be used for suggesting commonly preferred items to those within that group who have not yet purchased the item.
Union-Find / Connected Components / Weakly Connected	Finds groups of nodes where each node is reachable from any other node in the same group, regardless of the direction of relationships. It provides near constant-time (independent of input size) operations to add new groups, merge existing groups and determine whether two nodes are in the same group.	Union-Find / Connected Components is often used in conjunction with other algorithms, especially for high-performance grouping. As a preposing step for undirected graphs, it can help quickly identify disconnected groups.
Louvain Modularity	Measures the quality (i.e., presumed accuracy) of a community grouping by comparing its relationship density to a suitably defined random network. It's often used to evaluate the organization of complex networks, in particular, community hierarchies. It's also useful for initial data preprocessing in unsupervised machine learning.	Louvain is used to evaluate social structures in Twitter, LinkedIn and YouTube. It's used in fraud analytics to evaluate whether a group has just a few bad behaviors or is acting as a fraud ring that would be indicated by a higher relationship density than average. Louvain revealed a six-level customer hierarchy in a Belgian telecom network.

Table continued on next page

Optimized Graph Algorithms in Neo4j

It's time to reveal the meaning of the connections in your data—and make breakthrough discoveries about the forces that drive your disoveries.

Community Detection Algorithms - Continued

Algorithm Type	What It Does	Example Use
Local Clustering Coefficient / Node Clustering Coefficient	For a particular node, it quantifies how close its neighbors are to being a clique (every node is directly connected to every other node). For example, if all your friends knew each other directly, your local clustering coefficient would be 1. Small values for a cluster would indicate that although a grouping exists, the nodes are not tightly connected.	Local Cluster Coefficient is important to estimating resilience by understanding the likelihood of group coherence or fragmentation. Analysis of a European power grid using this method found that clusters with sparsely connected nodes were more resilient against widespread failures.
Triangle-Count and Average Clustering Coefficient	Measures how many nodes have triangles and the degree to which nodes tend to cluster together. The average clustering coefficient is 1 when there is a clique, and 0 when there a no connections. For the clustering coefficient to be meaningful it should be significantly higher than a version of the network where all of the relationships have been shuffled randomly.	The Average Clustering Coefficient is often used to estimate whether a network might exhibit "small-world" behaviors which are based on tightly knit clusters. It's also a factor for cluster stability and resiliency. Epidemiologists have used the average clustering coefficient to help predict various infection rates for different communities.

Use the Power of Connections to Drive Discoveries

The world is driven by connections. Neo4j Graph Analytics reveals the meaning of those connections using a practical, optimized graph algorithms built atop the world's leading graph platform.

To tap into the forces that drive discovery, visit *neo4j.com/graph-analytics* or contact us at *info@neo4j.com*. Let's do great things together.

Neo4j, Inc. is the graph company behind the #1 platform for connected data. The Neo4j graph platform helps organizations make sense of their data by revealing how people, processes and digital systems are interrelated. This connections-first approach powers intelligent applications tackling challenges such as artificial intelligence, fraud detection, real-time recommendations and master data.

The company boasts the world's largest dedicated investment in native graph technology, has amassed more than eight million downloads, and has a huge developer community deploying graph applications around the globe.

Questions about Neo4j?

Contact us:

1-855-636-4532

info@neo4j.com