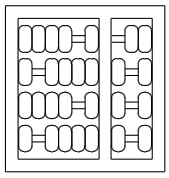

INSTITUTO DE COMPUTAÇÃO
UNIVERSIDADE ESTADUAL DE CAMPINAS

Querying complex data

L. Gomes-Jr *L. F. Costa* *A. Santanchè*

Technical Report - IC-13-27 - Relatório Técnico

October - 2013 - Outubro

Querying complex data

Luiz Gomes-Jr* Luciano da F. Costa André Santanchè

Abstract

Database technology has advanced to support increasingly complex data – from relations to semi-structured data and unstructured documents. More recently, graph databases have regained attention following demands from applications like social networks and recommendation systems. Graph analysis, usually associated with the Complex Networks field, has become a central tool in areas such as biology, physics and linguistics. Database management systems should improve support to these data and applications beyond the data model level tackled by current graph databases, including more flexible querying models and management mechanisms.

In this paper, we define the characteristics of the highly interconnected data that underlies many of these modern applications. We adopt the term *complex data* as a reference to the field of complex networks. A database management system for complex data requires a flexible query model that explores the topology of the relationships, taking into account their eventual uncertainty. Efficient query processing becomes a challenge, requiring new mechanisms for relationship-based query optimizations.

To meet the new requirements, our solution models complex data as property graphs with weighted relationships. We propose a new query language that allows ranking of elements based on properties of the topology of the graph. The queries are evaluated based on a variation of the spreading activation model, which is the core of the query processor and the main target for query optimization strategies. Experiments with real data show the practicability of our approach and support our analysis of several query optimization and approximation mechanisms.

1 Introduction

Data and query models have evolved towards supporting increasingly complex interrelationships. Highly structured models for management of relational data, with precise query semantics and predictable results, preceded semi-structured models that added path indirection and flexible schema. Document management and IR has to deal with even greater indirections, typically correlating imprecise queries (e.g. keywords) to lists of results ranked according to metrics that are unknown by the users, and correlating documents based on keyword similarities. More recently, graph databases have taken this diversity to new levels, allowing unrestricted correlation of data elements.

*Instituto de Computação, Universidade Estadual de Campinas, 13081-970 Campinas, SP. Work partially financed by the Microsoft Research FAPESP Virtual Institute (NavScales project), CNPq (MuZOO Project and PRONEX-FAPESP), INCT in Web Science (CNPq 557.128/2009-9), CAPES and FAPESP (Proc. 2012/15988-9)

As a consequence of the advances in models, algorithms and computing power, new types of applications became possible, such as social networks, recommendation systems and collaborative filtering. Network analysis, usually associated with the complex network field, has become an important resource paving the way to diverse applications such areas as systems biology, neuroscience, communications, transportation, power grids, and economics [12].

Data associated with these new applications are often more complex, ambiguous and less predictable than in previous settings. The analysis must rely on inferences based on how the data are correlated through the intricate networks formed by relationships among data elements. Typically, the topology formed by these relationships does not comply to a schema and may represent diverse types of associations with different strengths or weights – and richer types of links (e.g. involving categorical information) can also be considered.

Typical current DBMS's querying and management mechanisms are inadequate in supporting the new applications and handling the data created by them. Even graph databases assume an often unpractical level of structure for the underlying database: typical graph query languages are based on path patterns that assume that the graph topology is known, and do not take into account the strength or uncertainty of the relationships. Furthermore, correlating data based on complex properties of the topology of the graph is impractical.

Consider, for example, the following queries:

- retrieve candidate diagnosis relevant to a given patient based on her symptoms and correlations with other similar patients;
- retrieve documents related to the keyword query "US elections" and the topic *politics*, written by democrat journalists, ranked by relevance to the keyword query and reputation of the author;
- retrieve streets in a given city based on their relevance (specificity) to retail stores of sports goods and convenience of route in respect to my current location;
- retrieve fish species that play an important (influential) role in the marine food web of a given region and that are related to the biome 'coral reef'.

These types of queries have become commonplace in diverse application scenarios. However, there is still no query or management model that can seamlessly tackle all the underlying concepts. Answering these queries requires non-trivial analysis of the topology of the network structure in the underlying data. For example, answering the first query requires correlating symptoms and diagnoses, as well as data from patients in similar conditions. The analysis should also take into account that symptoms can be weakly or strongly correlated to diagnoses. The second query deals with associations from heterogeneous models. It requires the assessment of concepts like relevance and reputation, which are linked to the way the underlying data is interconnected (the topology formed by the relationships).

Supporting these queries in a DBMS brings several new architectural requirements: (i) the data model must support the high level of complexity, (ii) the query language should be flexible enough to allow correlation of data when little is known about how they are

linked and organized, (iii) a new abstraction for query evaluation should fully support the query language while allowing for under-the-hood optimizations, (iv) data management mechanisms must be coherent with the heightened importance and diversity of relationships.

Here we propose a new definition for the increasingly important type of data we aim at supporting. Inspired by the area of complex networks, we adopted the term *complex data*. This paper also aims at contributing to the specification of the requirements for a CDMS (Complex Data Management System). Our main focus here is on query processing and optimization issues.

Our solution is based on a property graph data model with weighted relationships. We propose a new query language that allows ranking of elements based on properties of the topology of the graph. The queries are evaluated based on a variation of the spreading activation model, which is the core of the query processor and the main target for query optimization strategies.

This paper is organized as follows: Section 2 describes related work. Section 3 presents the definition of *complex data* and the related challenges faced by a CDMS. Section 4 defines our proposed query model and shows examples of our query language. Section 5 describes experiments that assess diverse aspects of query processing and optimization. Finally, Section 6 concludes the paper.

2 Foundations and Related Work

The query model proposed here lies in the intersection of several related fields. The Complex Networks [12] field is an outstanding example of complex data manipulation. One of the main aspects of the field is analyzing highly interconnected data derived from diverse scenarios (e.g. social, biological or transportation networks). Current research in this area is based on graph models, employing global graph metrics to assess network properties that are typically rendered for visual analysis. Our proposal can be seen as offering new tools for the complex network analysis workflow. As far as we know, this is the first attempt to integrate the areas of complex networks and databases. We aim at providing adequate mechanisms to manage and query the associated data.

Graph databases are the best current option for storing and querying this type of highly correlated data. However, despite implementing the flexible graph model, other mechanisms in the database, such as querying and data management, are basically the same as in traditional relational databases. The declarative queries offered by most databases are limited to exact matches and assume the user knows how the data is organized. To implement more complex queries, users need to resort to imperative programming APIs, resulting in code that is hard to maintain and restricting opportunities for query optimization. Here we aim at enabling complex queries in a declarative model, with optimizations handled by the database.

A closely related research area, in terms of enabling more flexible querying for heterogeneous data models, encompasses the initiatives for integration of Databases and Information Retrieval. Following the initial identification of challenges and applications, several successful approaches have been proposed and implemented [15]. Most prominent research focuses

on keyword queries over structured data and documents and top-k ranking strategies.

Keyword query research draws from the simple yet effective keyword query model to allow integrated querying over documents and structured data. Most of the frameworks match keywords to documents, schema and data integrated in a graph structure. The connected matches form trees that are ranked based on variations of IR metrics such as tf*idf and PageRank. Some of the research focuses on optimizing the top-k query processing [10] while others implement more effective variations of the ranking metrics [11].

Keyword queries over structured data are intended for tasks where the schema is unknown to the user. The techniques are effective for data exploration, but there is no support for more principled interactions. There are conceptual and structural mismatches among queries, data and results that make returned matches hard to predict and interpret.

The research on top-k queries focus on enabling efficient processing of ranked queries on structured and semi-structured data. Ranking is based on scores derived from multiple predicates specified in the query. The main challenge is to compute results avoiding full computation of the expensive joins. The proposals vary on adopted query model, data access methods, implementation strategy, and assumptions on data and scoring functions (see [6] for a contextualized survey). Scoring functions enable ranking based on properties of data elements. There is, however, no simple means to rank results based on the context of elements or how they are correlated, which are central features of our scheme.

Less specific proposals, that also aim at increasing query expressiveness for interconnected data, have been developed especially in the context of knowledge bases. Kasneci et al. [7] propose a new querying model for knowledge bases generated from IE. The model allows pattern matching and *relatedness* queries that allow flexible correlation of elements in the base. Rodriguez et al. [14] propose a query model that allows the context of a user's knowledge base to influence query results. White and Smyth [16] present several algorithms to assess *importance* of nodes in a network. The calculation of the scores can be biased based on a set of initial nodes, which makes the approach more flexible than alternatives such as PageRank [1].

All these proposals aim at addressing the problem of inflexible query models that are inadequate to current application needs. However, the basic concepts of relatedness, context or importance are defined by the model and there are no means for users to express application-specific interpretations of the concepts. In our proposal, we offer declarative means for users to express combinations of metrics that can be adapted to specific information needs. Our proposal aims at scenarios where relationships among data elements can be uncertain and there is no underlying schema. Nevertheless, relationship analysis is central to understanding the data. Based on a more flexible query model, our goal is to propose query processing and data management mechanisms that are adequate to these complex settings.

Graph query languages are typically concerned with subgraph matching. Query processing in this scenario must tackle optimization issues related to graph isomorphism and similarity [9]. Our language model adds expressiveness to this type of queries, allowing flexible ranking based on topology-aware correlation metrics applied to matched elements. Although graph isomorphism and similarity strategies are relevant to many of our query scenarios, we rely on the efficient implementations already available on standard graph

databases (which have provided adequate performance), and focus on optimization strategies for our new layer of expressiveness.

3 Complex data definition and challenges

Complex data is characterized when relationships are central to data analysis. In these cases, the graph formed by data entities (nodes) and relationships (links) present properties typical of complex networks. In a complex network [12], the patterns defined by the interconnections are non-trivial, deviating substantially from cases where connections have the same probability (e.g. lattices or random graphs).

This heightened importance of relationships is evident in several areas, such as social networks, topological maps, gene-protein interaction networks, brain network models etc. These are all complex structures that requires specific techniques for analysis. In all cases, relationship analysis is a major aspect of the understanding. Typically, these structures generate emergent behavior, which are determined by the complex interactions among their simple constituent elements.

Relationships may also be uncertain or ambiguous. The relationships among data elements can represent associations with variable strength or be inherently uncertain depending on the nature of the systems they model. Examples of this type of phenomenon are plentiful: Protein interaction networks are based on co-occurrence observations that could in reality be associated with causal, indirect, or incidental correlation; the 'friendship' relationship in social networks is highly ambiguous, meaning different things for different people; entity recognition over text is uncertain, so the algorithms usually generate associations with a confidence value.

As a result of increased capacity of data storage and processing, these scenarios have come forth in other areas, such as enterprise data management. A typical institution nowadays stores and processes many textual documents alongside traditional structured data, communication and transaction records, and fast changing data about market and competition. These data are highly interlinked, by design or through intricate (and potentially imprecise) data analysis procedures such as named entity recognition, sentiment analysis, and recommendation systems.

Query processing and data management face several challenges in these scenarios. Precise answers are impractical due to the exponential computational costs and the imprecision of the data; analysis of the relationships must be tolerant of the inherent uncertainty; query optimization cannot depend on an underlying schema; and there should be appropriate abstractions for managers to handle the data. In the next section we list the requirements for a Complex Data Management System (CDMS).

3.1 Requirements for complex data querying and management

(i) The *data model* must support a high level of complexity. The data in target applications typically do not comply to pre-defined schemas. The high number and diversity of the relationships require a model where relationships are first-class citizens. Graph models are obvious choices in these settings. Their flexible modeling characteristics enable easy

mapping of most types of data. Nodes with immediate access to neighbors is also an important feature for the type of computation involved. Here we adopt weighted edge-labeled property multigraphs (Section 4.1) to encode complex data.

(ii) The *query language* should be flexible enough to allow correlation of data when little is known about how they are linked and organized. The language should also allow for query processing optimizations or approximations, key features when dealing with computing intensive complex data. In our research, we developed a declarative query language that extends existing graph languages by introducing ranking based on a set of flexible correlation metrics. We introduce the language in Section 4.4.

(iii) New *abstractions for query evaluation* should fully support the query language while allowing for under-the-hood optimizations. Abstractions such as joins over relational data or path predicates over semi-structured data are inadequate for the often loose and indirect correlations among data elements in the schema-less scenario. Moreover, query processing algorithms must exploit confidence values to compute answers. Data analysis must take advantage of the broader context of the relationships, leveraging the amount of concordance among them to assess more reliable associations. Query optimization is also a challenge, since reliable schema metadata is usually the basis of efficient query processing. Here we adopt a variation of the spreading activation model as our main abstraction for query evaluation. The model allows the specification of the ranking metrics that are the basis of our query language. In Section 5 we present several experiments on parameter tuning, query optimization and approximations for our model.

(iv) The *data management mechanisms* such as Data Manipulation Languages (DMLs), indexes and collection of data statistics must be coherent with the heightened importance and diversity of relationships. In our framework, we introduce the concept of *mappers* in our DML, which are similar to stored procedures, but are aimed at relationship creation and can be integrated in the query language to allow query-time correlation of data. Mappers and data management issues will not be further discussed in this paper (an overview can be found in [4]).

4 Complex data querying

In this section we present our proposals for data model, query processing, and query language for complex data.

4.1 Data model

We adopt the graph data model to represent complex data. The graph model has relationships as first class citizens, allowing flexible representations of a wide range of real-world data [13]. More specifically, we adopt a labeled weighted multidigraph model.

A labeled weighted multidigraph G is a directed multigraph with weighted edges and labeled vertices and edges. Formally, $G = (V, E, \Sigma_V, \Sigma_E, s_E, t_E, \ell_V, \ell_E, w)$ where V is a set of vertices and E is a set of edges, Σ_V and Σ_E are finite alphabets for the available vertex and edge labels, $s_E : E \to V$ and $t_E : E \to V$ are maps indicating the source and target vertex of an edge, ℓ_V and ℓ_E are maps describing the labeling of the vertices and edges,

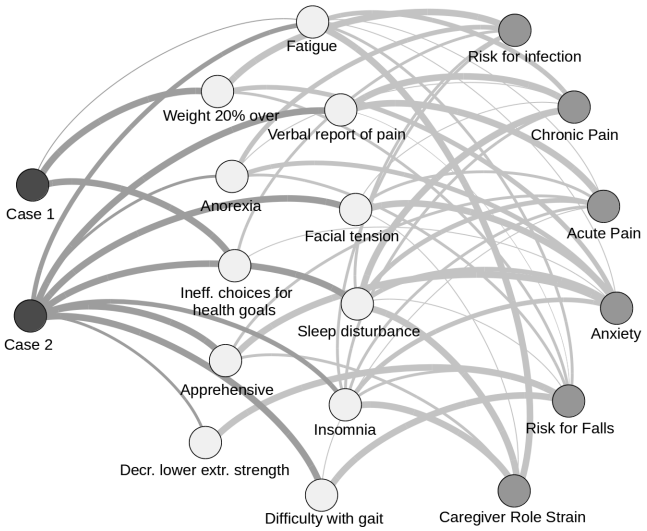

Figure 1: Sample subgraph of a dataset for the nursing diagnosis (relationship weights are proportional to edge thickness, directions are omitted)

and $w : E \to (0, 1]$ maps edges to their weights. In practice, our data model is also a property graph [13], adding attributes for data management convenience, but since it does not impact the definition of our query model, we omit these details.

Figure 1 shows an example of the type of graph data complying to our definition. This definition (supported or adaptable in several graph databases) is appropriate to represent complex data. The simplicity of the model and the support for unrestricted correlation of data elements are the main reasons for adoption. Other models (e.g. RDF) with equivalent flexibility could also be employed. More details on the mapping of relational or unstructured data into such graph models can be found in [4].

4.2 Targeted Spreading Activation Model

Spreading Activation (SA) processes [2] were developed to infer relationships among nodes in associative networks. The mechanism is based on traversing the network from an initial set of nodes, activating new nodes until certain stop conditions are reached. By controlling several aspects related to this activation flow, it is possible to infer and quantify the relationships of the initial nodes to the reached ones.

This simple model has the fundamental requirements for the type of correlations we want to provide for complex data:

(i) it can derive correlations among any two sets of initial nodes and destination nodes; This is important to enable modeling of several correlation metrics, as described in Section 4.3.

(ii) the final value of the correlations decreases as the length of the contributing paths grows; This reflects the intuitive perception that closer elements are more correlated. The model allows tuning of this characteristic through a parameter for potential degradation.

(iii) the degradation of the potential imposes boundaries to query processing;

(iv) it can be implemented as graph traversal patterns [13]; The processing of these patterns are centered in origin nodes, resulting in localized processing. The computation of these patterns requires less memory than global ranking metrics such as PageRank and HITS. This type of computation is supported by several graph database systems[1].

We tweak the basic SA model by adding mechanisms to (i) adapt the process to the labeled graph model used, (ii) consider relationship weights, (iii) add a more strict and predictable termination condition, and (iv) make the process aware of the target elements. The last point is key to the semantics of the SA process for querying complex data and also to improve optimization opportunities (Section 5). We named the proposed SA variation as Targeted Spreading Activation (TSA).

The SA model used here is defined by the parameters G, N, I, O, a, t, d, c, l, and dir described, alongside other definitions, in Table 1. A SA process starts with origin nodes initially activated with potential a. Output potentials for each subsequent node are calculated by the function O. The output potential is spread through all relationships whose labels are in l that follow directions in dir. The potential for the reached nodes is calculated by function I. For the next iteration, the potential is spread to subsequent nodes, restarting the process, as long as the potential for reached nodes is higher than t and the number of iterations is lower than c.

Although simple in its definition, this is a very expressive model to build flexible correlation metrics. By specifying appropriate parameters and combining subsequent executions of TSAs, it is possible to define metrics that encompass concepts like relevance, reputation and similarity (Section 4.3). These metrics can be integrated in a declarative language with applications to a wide range of modern querying scenarios (Section 4.4).

Being the core of the querying process mechanism, the TSA process becomes the main target for query optimization strategies. Like with any other data or processing model, the practicability of TSA-based querying depends on architectural mechanisms to support data access optimizations and heuristics to provide approximate answers. These are the main challenges addressed in this paper (Section 5).

4.3 TSA-based ranking metrics

The TSA model can be the basis for the specification of ranking metrics for data correlation based on properties of the topology of the underlying graph. Here we focus on the metrics of relevance and connectivity. The definitions and usage of other metrics (Reputation, Influence, Similarity, and Context) can be found in [3].

Def. 1. $relevance(m, n) = SA(m)_n$,
with $O(n) = \dfrac{I(n) * d}{|sub(n)|}$

Relevance between two nodes is a measure that encompasses correlation and specificity. Correlation is proportional to the number of paths linking the two nodes and inversely proportional to the length of the paths. Specificity favors more discriminative paths (i.e

[1] A good overview of applications and systems can be found in http://markorodriguez.com/2013/01/09/on-graph-computing/

paths with fewer ramifications. It is easy to observe that this definition resembles the definition of relevance between queries and documents in a information retrieval setting. Traditional tf*idf term weighting can be readily emulated in our scheme when terms, queries and documents become nodes of a graph. Our definition is, however, a generalization of the concept that can be applied to any type of graph data and with any number or type of relationships in between m and n.

Def. 2. $rrelevance(m, n) = SA(m)_n + \overline{SA(n)_m}$,
with $O(n) = \dfrac{I(n) * d}{|sub(n)|}$

Reciprocal Relevance (RRelevance) between two nodes aggregates the relevance in both directions. In an information retrieval setting, it would be equivalent to aggregating document size normalization to the relevance model.

Def. 3. $connectivity(m, n) = SA(m)_n$

Connectivity between two nodes is a measure that assesses how interconnected the nodes are. The score is proportional to the number of paths linking the nodes in the network activated by the SA algorithm.

The definitions and usage of other metrics (Reputation, Influence, Similarity, and Context) can be found in [3].

4.4 Querying complex data

Having the ranking metrics interpreted as graph analysis tasks, it is possible to integrate them in a declarative query language. As opposed to creating an entirely new query language, we decided to leverage existing languages by defining an extension language.

A convenient way to integrate the ranking metrics into existing query languages is to add a "RANK BY" clause. The clause should enable an arbitrary combination of metrics that expresses the global ranking condition defined by the user. We encode the clause in the extension query language that we denominated in* (or in star). in* can be used to extend other languages, for example, extended SPARQL becomes inSPARQL by convention. This strategy is a good fit for graph languages with SQL-inspired syntaxes, such as SPARQL[2] and Cypher[3]. A similar strategy could be developed to other types of languages.

We illustrate the use of our query language through examples in inSPARQL and inCypher. These queries are meant to demonstrate the expressiveness of the approach in a wide range of applications.

Figure 2a shows an extended SPARQL query that retrieves actors whose careers are strongly correlated with the director Woody Allen (id 8501). Details about the interpretation and processing of this query are presented in Section 5.

Figure 2b shows a Cypher query that suggests diagnoses of patients based on their symptoms. It is based on an underlying database containing symptom-diagnosis relationships specified by professionals in a previously unrelated project. Figure 1 shows a subgraph of this database. The combination of the Connectivity and Relevance metrics showed results compared to unassisted diagnoses from health professionals.

[2] http://www.w3.org/TR/sparql11-query
[3] http://docs.neo4j.org

a
```
SELECT DISTINCT ?actor WHERE {
  ?film movie:director director:8501 .
  ?film movie:actor ?actor .
  ?film movie:initial_release_date ?date .
  FILTER ( fn:starts-with(?date, "199") ) }
RANK BY RELEVANCE OF ?actor TO director:8501
```

b
```
START diag=node(*)
   WHERE has(diag.Type) and diag.Type = "Diagnose"
   RETURN diag
RANK BY
   1 RELEVANCE OF diag TO node($patient) ,
   2 CONNECTIVITY OF diag TO node($patient)
```

c
```
SELECT DISTINCT ?product
 WHERE { ?product :type :Product .
   FILTER NOT EXISTS (:bob :purchased ?product) }
RANK BY RELEVANCE OF ?product TO :bob
 FOLLOW (:friendsWith, :purchased)
 DEPTH 3 DIRECTION BOTH
```

d
```
SELECT ?species
 WHERE { ?species :type :MarineSpecies }
RANK BY
 3 INFLUENCE OF ?species FOLLOW :preysOn
    DIRECTION OUTBOUND,
 1 RELEVANCE OF ?species TO :CoralReefBiome
```

Figure 2: Examples of extended queries (namespaces have been omitted)

Figure 2c shows a product recommendation query (SPARQL) that finds products that the client Bob (with uri :bob) has not purchased. The query traverses Bob's friendship network to find products purchased by his friends that might be relevant to him. The spreading activation interpretation of this query evaluation also implies that products purchased by Bob, even though they do not appear in the results, will be traversed on the way to customers that have co-purchased these products, which in turn will activate other products from these customers.

The query in Figure 2d (SPARQL) ranks species that play an important role in the food web and are related to the biome of coral reefs. This type of query would identify species that should be main targets for monitoring and preservation efforts.

5 Query processing and optimization

In this section we show experiments that demonstrate aspects of query execution and options for query optimization. We focus on reports for the RRelevance and Connectivity metrics, which we consider good representatives of the model because of their (i) applicability in many areas, (ii) cognitive appeal, and (iii) challenges for query processing.

The database used in the experiments is the Linked Movie Data Base (LinkedMDB) [5], which we think is a good representative for the type of data we aim at. The database integrates data from several sources (FreeBase, OMDB, DBpedia, Geonames, etc). The database contains 3,579,616 triples. The database represents the bulk of the relevant production in a real and important area of human activity, demonstrating that our framework

can be applied to real scenarios.

The query used in the experiments is shown in Figure 2a. The query combines graph pattern matching and structured filtering (equivalent to selection in relational algebra) predicates and our new RANK BY clause. The query returns actors that acted in films directed by Woody Allen in the 90's. The results are ranked by relevance of the actors to Woody Allen (director). This query should be interpreted as raking actors according to how linked to the director their careers are – a common pattern throughout Allen's idiosyncratic production.

5.1 Baseline

As the baseline for the performance and accuracy experiments, we executed the query in Figure 2a with parameters [t=0.1, d=0.9, c=2, a=100]. These are conservative parameters that showed good results in our analysis. By conservative we mean high values for d and a, and low values for t and c. As should be clear from the experiment data, any combination of these parameters that follow this conservative rule of thumb would produce similar results. Parameter c, which has the biggest impact in performance predictability, was set at 2 because this is the length of the path for the most important relationships between actors and directors (actors are linked to directors through common films). The next section shows how varying the parameters affects performance.

The top-10 and bottom-10 ranked actors are shown in Table 2. The top ranked actor is Woody Allen himself[4]. Allen is well known for interpreting roles in his films, and he rarely performs in films from other directors. Mia Farrow, the second highest rank, has her career strongly linked to the director, acting in 13 of Allen's films, out of her total of 39 films registered in the database.

Less known actors also appear in the top-10 list. Hazelle Goodman, for example, has only one performance recorded in the database, which would make her career highly linked to Woody Allen.

Low ranking actors are usually actors that participated in many films but few of them were directed by Woody Allen. This is the case for Uma Thurman and Robin Williams, for example, who each perform in only one of Allen's films. The interpretation is that low ranked actors would not have their careers linked to Woody Allen, despite having been cast in his movies.

The analysis of the query results reaffirms that the graph interpretation of relevance proposed here is indeed strongly correlated to the typical interpretation of relevance (e.g. in information retrieval applications). We ran the experiments in a regular desktop machine that was querying a Neo4j embedded database. There was no database server involved, so executions are cold by definition (no usage of database caching mechanisms). Execution time numbers reported are averaged over 3 runs for each variation (in a total of 60 runs).

[4]LinkedMDB uses distinct descriptors for the actor and the director, implying that they are separate entities.

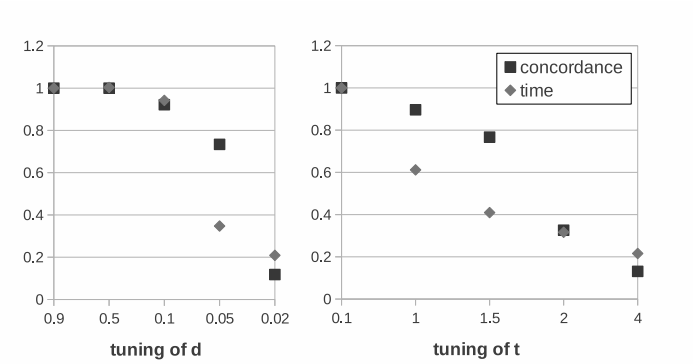

Figure 3: Correlation and normalized execution time for variation of parameters d and t (in respect to the baseline)

5.2 Parameter tuning

The parameters in our SA model ultimately determine how far the activation process would go in its exploration of the graph. This has consequences in terms of performance and completeness of query execution. Relaxed values for the parameters, which would allow bigger portions of the graph to be included in the query, have expensive computational requirements, but render more contextualized ranking that might include non-obvious aspects of the correlation of the elements in the query. The best balance between performance and completeness is application-specific. The experiments here are meant to show the impacts of parameter tuning on the execution time and accuracy.

Decay factor (d): Figure 3 (left) shows the values for normalized execution time and correlation for activation thresholds varying from 0.9 to 0.02. To assess correlation between the produced ranks and the baseline, we used the Kendall tau rank correlation coefficient [8]. The graph shows how both execution time and correlation drops as we use more aggressive values for d. Lower values for d mean faster degrading of the activation potential, which implies more effective pruning of the expanding activated network.

Activation threshold (t): Figure 3 (right) shows the values for normalized execution time and correlation for decay factors varying from 0.1 to 4. As with the decay factor, correlations falls at rates proportional to performance gains. Appropriate values might be set according to query-specific tolerance for inaccuracies and performance requirements.

Depth (c): c is the parameter whose impact in performance is most predictable, since it directly limits the growth of the diameter of the activated network. Figure 4 (top curve) shows how execution time increases sharply as c grows. The correlation between the ranking, however, stays high: correlation between c=2 and c=3 is 0.97, dropping to 0.89 for c=5.

The graph shows that the sharp increase in execution time is slowed down by the self-containing characteristic of the SA algorithm. More aggressive parameter tuning would of course increase this effect. For example, setting t=1 reduces execution time to less than half for c=5 (maintaining a similarly high correlation).

5.3 TSA optimization

The expressiveness of the queries allowed by the extended languages sometimes blurs the line between declarative queries and data analysis. Even though some tasks would not present tight response time constraints, it is clear that these queries can require, in typical settings, traversing massive numbers of nodes. It is therefore important to introduce optimization mechanisms and aggressive heuristics to compute approximate answers. We now show some directions we are taking for querying optimization in the framework.

Retracing TSA: For metrics like RRelevance, which executes two TSA processes in reversed directions, the query processor can take advantage of the fact that the first execution will find all possible paths between the nodes. The second execution can retrace the network derived by the first process, avoiding exploration of paths that do no contribute to the final score. Execution times for this strategy are reported in Figure 4 as rTSA.

TSA transformations: In some cases, it is possible to invert the source and destination nodes in a TSA process. This can lead to reduced execution times when favoring nodes with smaller degrees as starting points for the process. This has another more important consequence: by choosing the best starting point, it is possible to reuse the results from the first SA process, avoiding duplicate computations.

Inverting the TSA without affecting the ranking is only possible in some cases. The possible transformations for the RRelevance and Connectivity metrics are listed below.

Transformation 1. $rrelevance(m,n) = \overline{SA(n)_m} + SA(m)_n$

Transformation 2. $connectivity(m,n) = \overline{SA(n)_m}$

Consider, for example, the baseline query using the *Connectivity* metric. Applying the TSA transformation 2 reduces execution time to about half. Transformation 1 changes the order of the TSA processes, which becomes important when considering caching (next item).

We are investigating other rank preserving TSA transformations. Specifically, more general transformations could be developed for TSAs whose activation network forms DAGs (directed acyclic graphs). Even for cases when ranking preservation is not possible, these transformations would still be a good approximation heuristic. A simple heuristic to determine whether a transformation rule should be applied is based on the collective degrees of starting nodes.

TSA Caching: Frequently, the execution of queries in in* will demand the execution of several TSA processes with the same origin node. Furthermore, repetitions of origin nodes can be achieved through transformations. Caching the results for destination nodes allow a subsequent TSA process to reuse the score calculated by the previous process. Execution times for a combination of caching, retracing and transformations is reported in Figure 4 as cTSA.

5.4 TSA approximation

In settings that do not require precise answers, processing time in TSA-based querying can also be reduced through result approximations. In this section we describe some alternatives and experimental results for the ones currently deployed.

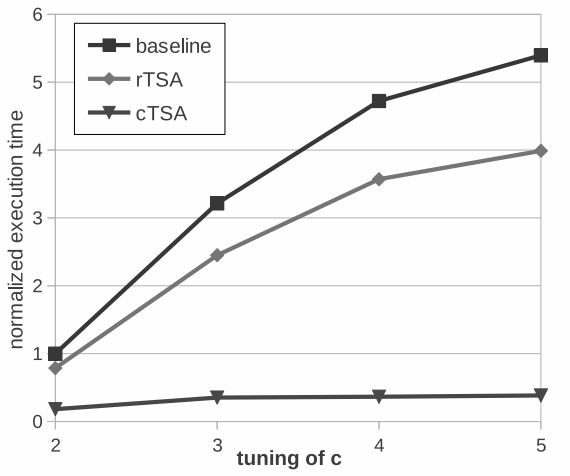

Figure 4: Normalized execution time for optimization strategies with variations of parameters c

Restricting traversal edges and direction: The query in the baseline explores recursively all edges in all directions emanating from the actor nodes. Considering the substantial size and complexity of the database, processing the query implies reading a large number of nodes. More focused queries are, however, allowed by the language. By setting the modifiers FOLLOW and DIRECTION, the user can restrict the paths taken by the spreading activation process. For example, setting FOLLOW to ['linkedmdb:actor', 'foaf:made'], which are the labels for movies' actors and directors, reduces query execution time by more than one order of magnitude (10.77 fold). Setting the direction to INBOUND alone reduces query time even more (11.25 fold). Setting both modifiers decreases execution time 13.5 fold.

One heuristic for query execution could choose edges and directions based on statistics to rewrite queries. It is important to emphasize that the new queries would be semantically different from the original. For example, the semantics of the query in the baseline is that of relevance between actors and Allen based on any possible relationship sequence present in the database. Heuristics from query rewriting should therefore favor performance while also maintaining a good amount of contextual information.

Shortest Paths TSA: The network of activated nodes in a SA process often grows unnecessarily big by exploring paths that do not lead to destination nodes. One way to avoid this is to restrict the activation process to paths that are known to lead to destination nodes. Current graph databases implement fast algorithms for calculating shortest paths, which can be exploited by a variation of the TSA. An implementation of this strategy over the Neo4j database reduced the execution time for the baseline 6 fold. As previously, this heuristic reduces the amount of contextual information in the final scores, affecting the ranking.

Random Walker-based Sampling TSA: In some cases, the databases will be too big or too complex for exact computations of the metrics in the queries. For these scenarios,

we have developed a variation of the TSA process based on random walkers. In these settings, a random walker would randomly choose relationships to follow in the execution of the TSA process. We are now working on a variation of this implementation that would enable random walkers to share information about the network, consequently biasing their navigation towards relationships that are found to lead to destination nodes.

We expect this type of processing to be a key enabler of CDMS in BigData scenarios. Proper tuning of parameters for the random walkers could enable control over the balancing between efficiency and contextualization of the TSA processes.

Parallelization and distribution: TSA processing can be easily parallelized in a multicore CPU. In cases that do not require precise answers, synchronization could be avoided to further improve performance. The underlying processing model for TSA can also be ported to distributed graph databases, improving the scalability of the processes.

5.5 Discussion

The experiments indicate that our approach is practical in terms of performance and allow for several optimization and approximation mechanisms. Average execution times for the metric in the baseline query took under 1.5 seconds, which was reduced to 0.28 seconds after score preserving optimizations. We think this would already be an interesting achievement for the non-trivial query, database, and computations we are dealing with, but there is room for radical improvements.

To simplify implementation and facilitate incremental design of our query processor, we are employing a tall stack of APIs[5] that can be reduced for further performance gains.

Appropriate tuning of the TSA parameters and selection of the best processing strategy is related to diverse properties of the graph to be traversed. The most important variables are (i) the expected degrees for source and destination nodes, (ii) the expected average path length between the source and the destination nodes, and the (iii) expected branching factor in the paths. These variables can be calculated or estimated by the query processor based on statistics sampled from the database, allowing automatic optimization of query processing. We are working on defining the important statistics to maintain and on designing heuristics for this automation.

6 Conclusion

New applications and demands from several areas require the analysis of intricately interrelated data. Following a similar trend, data managed by institutions are becoming increasingly complex. Expressive querying and adequate management of this type of data presents several challenges to current database technology.

In this paper we define the requirements associated with the new challenges. We propose several mechanisms for query specification and query processing optimization. Our query model redefines metrics that rank entities based on the topology of their correlations. As

[5] our implementation stack includes the graph traversal language Gremlin, the Blueprints and OpenRDF frameworks, Neo4j and the Virtuoso triple store

suggested by the query examples presented (Figure 2), it is possible to represent information needs that would require a level of data analysis that is beyond current implementations of typical database systems.

Our experiments show that our approach is practical in terms of performance and that our language can express complex concepts in real data and application scenarios. Combining the TSA model and a declarative query language offers many opportunities for query optimization. Enabling this level of optimization based on typical graph querying mechanisms would require complex ad-hoc solutions.

Ongoing work is focused on improving the modularity of the system, further implementation and test of all the proposed metrics, development of new optimization strategies, and expansion of application scenarios. Future work will include time analysis and dynamic networks support, schema-free materialized views, and workload-based collection of statistics to support query optimization.

References

[1] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1-7):107–117, 1998.

[2] F. Crestani. Application of spreading activation techniques in information retrieval. *Artif. Intell. Rev*, 11(6):453–482, 1997.

[3] L. Gomes-Jr, R. Jensen, and A. Santanchè. Towards query model integration: topology-aware, ir-inspired metrics for declarative graph querying. In *Second International Workshop on Querying Graph Structured Data (GraphQ-EDBT)*, 2013.

[4] L. Gomes-Jr and A. Santanchè. The Web Within: leveraging Web standards and graph analysis to enable application-level integration of institutional data. Technical Report IC-13-01, Institute of Computing, University of Campinas, January 2013.

[5] O. Hassanzadeh and M. Consens. Linked movie data base. In *Proceedings of the 2nd Workshop on Linked Data on the Web (LDOW2009)*, 2009.

[6] I. F. Ilyas, G. Beskales, and M. A. Soliman. A survey of top-k query processing techniques in relational database systems. *ACM Computing Surveys*, 40(4):11:1–11:58, Oct. 2008.

[7] G. Kasneci, F. M. Suchanek, G. Ifrim, M. Ramanath, and G. Weikum. NAGA: Searching and Ranking Knowledge. In *2008 IEEE 24th International Conference on Data Engineering*, pages 953–962. IEEE, Apr. 2008.

[8] M. G. Kendall. A new measure of rank correlation. *Biometrika*, 30:81–93, 1938.

[9] A. Khan, N. Li, X. Yan, Z. Guan, S. Chakraborty, and S. Tao. Neighborhood based fast graph search in large networks. In *SIGMOD*, pages 901–912. ACM, 2011.

[10] Kimelfeld and Sagiv. Finding and approximating top-k answers in keyword proximity search. In *PODS: 25th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, 2006.

[11] Y. Luo, W. W. 0011, X. Lin, X. Zhou, J. W. 0001, and K. Li. SPARK2: Top-k keyword query in relational databases. *IEEE Trans. Knowl. Data Eng*, 23(12):1763–1780, 2011.

[12] M. Newman. The structure and function of complex networks. *SIREV: SIAM Review*, 45, 2003.

[13] M. A. Rodriguez and P. Neubauer. The graph traversal pattern. *CoRR*, abs/1004.1001, 2010.

[14] M. A. Rodriguez, A. Pepe, and J. Shinavier. The Dilated Triple. In *Emergent Web Intelligence: Advanced Semantic Technologies*, pages 3–16. Springer London, June 2010.

[15] G. Weikum, G. Kasneci, M. Ramanath, and F. Suchanek. Database and information-retrieval methods for knowledge discovery. *Communications of the ACM*, 52(4):56–64, Apr. 2009.

[16] S. White and P. Smyth. Algorithms for estimating relative importance in networks. In *SIGKDD*, 2003.

notation	description
$SA(m)_n$	value for the potential of node n after the execution of the spread activation with initial activated node m (other parameters are omitted for brevity)
a, t, d, c	respectively, initial activation potential, firing threshold, decay factor, maximum number of iterations (depth)
l	set of labels that determine valid nodes for traversal
dir	$dir \subset \{inbound, outbound\}$; set of directions for traversal
$\overline{dir}$	$\overline{dir \cap \{inbound, outbound\}}$; reversed directions of dir
$\overline{SA(m)_n}$	equals $SA(m)_n$ with reversed directions, i.e. $dir \leftarrow \overline{dir}$
$I(n)$	function that calculates the input potential of a node. $I(n) = \sum_{m \in ant(n)} O(i) * w(m, n)$ in the default case (not supporting the multigraph definition for brevity)
$O(n)$	function that calculates the output potential of a node. $O(n) = I(n) * d$ in the default case
$ant(n)$	set of antecedent nodes, i.e. nodes linked to n through relationships in l that follow the directions in dir
$sub(n)$	set of subsequent nodes, i.e. nodes linked to n through relationships in l that follow the directions in $\overline{dir}$

Table 1: Notation used in the definitions

top 10	name	bottom 10	name
1.00	Woody Allen	0.02	Demi Moore
0.42	Mia Farrow	0.02	John Malkovich
0.24	Tony Darrow	0.02	Dom DeLuise
0.21	Julie Kavner	0.02	Stanley Tucci
0.21	Diane Keaton	0.02	Samantha Morton
0.16	Brian Markinson	0.02	Donald Pleasence
0.14	Dianne Wiest	0.02	Anthony LaPaglia
0.13	Hazelle Goodman	0.02	Sean Penn
0.12	Judy Davis	0.01	Robin Williams
0.11	Alan Alda	0.01	Uma Thurman

Table 2: Top-10 and bottom-10 ranked results for the baseline query (total of 98 returned actors)