GExp: Cost-aware Graph Exploration with Keywords

Mohammad Hossein Namaki, Yinghui Wu, Xin Zhang
Washington State University
{mnamaki, yinghui, xzhang2}@eecs.wsu.edu

ABSTRACT

We demonstrate GExp, an interactive graph exploration tool that uses keywords to support effective access and exploration of large graphs. GExp interleaves keyword query suggestion, which generates keyword queries that expand the original query, and query evaluation, that returns the answers to suggested queries for feedback. It advocates (1) cost-aware exploration, which suggests keyword queries that have low answer cost (thus high answer quality), and (2) incremental query evaluation to update the query answers with a bounded time cost. It differs from prior systems in its ability to identify and leverage substructures that augment original answers for query expansion and evaluation with provable cost bounds. A unique feature of GExp is that users can trade the quality of query answers with their evaluation cost by tuning the bound of answer cost in an ad-hoc manner. We demonstrate how GExp supports graph exploratory with three established keyword query classes with bounded time cost, and guarantees on result quality, using real-world knowledge bases and information networks.

ACM Reference Format:

Mohammad Hossein Namaki, Yinghui Wu, Xin Zhang. 2018. GExp: Cost-aware Graph Exploration with Keywords. In *SIGMOD'18: 2018 International Conference on Management of Data, June 10-15, 2018, Houston, TX, USA*. ACM, New York, NY, USA, 4 pages. <https://doi.org/10.1145/3183713.3193550>

1 INTRODUCTION

Graph search is challenging due to the nontrivial task of query construction. It is desirable to develop *exploratory* methods that make use of established user-friendly querying techniques. Beyond conventional “query-response” paradigms, graph exploration interleaves query expansion [5] and query evaluation [10] in a search session. Each session suggests queries, receives feedback from users on queries and results, and re-evaluates new queries.

Keyword queries (KWS) have been studied to support user-friendly graph search. Given a keyword query Q (a set of terms) and a graph G , KWS returns a list of answers $Q(G)$, where each answer G_Q is a subgraph of G that contains a set of *content nodes* that match the terms in Q . Specifically, G_Q can be a minimal rooted tree, weighted Steiner tree, or r -clique (nodes with pairwise distance bounded by a threshold r), for a query Q that is distinct root-based (DR) [15], Steiner tree-based (ST) [6], or Steiner graph-based (SG) [10], respectively. The *answer cost* of G_Q is determined

by aggregating the distance among its (content) nodes. Answers with low cost are preferred, following a common intuition that they “explain” the connections among content nodes with fewer edges.

Can we enable graph exploration with established KWS methods?

This is desirable but not easy. Without prior knowledge, users often need to revise the keywords as well as query semantics multiple times to get good answers that are relevant and with low answer cost. Moreover, in general, KWS over graphs is expensive. For example, evaluating ST queries are NP-hard [6]. This leads to long response and waiting time during exploration sessions.

GExp. We propose to enable cost-aware exploration over large graphs. We demonstrate GExp, a user-friendly tool for graph exploration with keywords. GExp has the following unique, desirable features that differ from conventional graph querying systems.

(1) *Cost-aware exploration.* GExp explores graphs with queries that have low-cost answers and low evaluation cost. Users only need to specify and tune a threshold that characterizes expected answer cost. Accordingly, GExp guarantees to suggest expanded queries, by adding diversified new terms, where each query is *guaranteed* to have non-empty answers with bounded answer cost. That is, GExp avoids suggesting “bad” queries that lead to high answer cost. It computes the query expansions by constructing augmented subgraphs of $Q(G)$, without performing expensive query evaluation.

Such cost-awareness makes graph exploration more controllable and reliable to end users, who use conventional KWS as in search engines, and only need to specify a simple threshold. Existing exploration tools (e.g., [3]) do not support such feature.

(2) *Incremental Optimization.* GExp extends incremental computation [7, 11, 12] to KWS with new optimality guarantees for established KWS query classes. Given a query Q , answer $Q(G)$, and an expanded query Q' , GExp incrementally updates $Q(G)$ to approximate answers $Q'(G)$ with a bounded time cost. This ensures the fast response of search sessions when exploring large G .

(3) *Ease of use.* GExp provides a simple configuration panel for users to tune the cost-aware exploration. Moreover, GExp provides built-in libraries of common information retrieval metrics, KWS query classes, and query evaluation algorithms to facilitate graph exploration with various query and cost models.

As a proof of concept, a primitive version of GExp is available [2]. Below we give an overview of GExp (Section 2), presenting cost-aware query expansion (Section 2.2), incremental query evaluation (Section 2.3) and architecture (Section 2.4). We will demonstrate each component of GExp and show how it supports cost-aware graph exploration with established KWS query classes (Section 3). We will also demonstrate use cases of GExp in knowledge base exploration and social recommendation.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

SIGMOD'18, June 10-15, 2018, Houston, TX, USA

© 2018 Association for Computing Machinery.

ACM ISBN 978-1-4503-4703-7/18/06...\$15.00

<https://doi.org/10.1145/3183713.3193550>

2 SYSTEM OVERVIEW

2.1 Workflow of GExp

Graph exploration in GExp is an interactive sequence of search sessions. Each session takes as input a pair $(Q, Q(G))$ that contains a KWS query Q and its answers $Q(G)$ (validated by users in the last session), and returns a new pair $(Q', Q'(G))$ that contains a new query Q' and answers $Q'(G)$, to be validated in the next session.

Sessions. GExp executes two building-block tasks in each session.

Cost-aware query expansion. Upon receiving a pair $(Q, Q(G))$ (Step 1 in Fig. 2), GExp computes a set of new queries Q' , where each query $Q' \in Q'$ expands Q with a set of new terms. GExp does not generate queries that have no answers with a bounded cost, specified by a user-defined, tunable answer cost threshold. It then returns Q' to users for validation (Step 2). Users can select suggested queries or update Q with their own terms.

Incremental query evaluation. Upon receiving a specified query Q' , along with original query Q and answers $Q(G)$ (Step 3), GExp computes the answers $Q'(G)$ by incrementally updating $Q(G)$, without re-evaluating Q' from scratch. It returns the pair $(Q', Q'(G))$ for validation (Step 4), or directly triggers the next session.

For each session, users can specify ad-hoc query class and alter cost threshold to alter the way of graph exploration.

Query classes. GExp currently supports cost-aware graph exploration with three established KWS query classes.

- **Distinct root-based queries (DR)** [9] search for trees with distinct root and bounded depth r . The answer cost of a tree G_Q , determined by a cost function $F(G_Q)$, is computed as the sum of distances from each content node to its root v_r .
- **Steiner tree-based queries (ST)** [6] computes minimum weighted Steiner trees with bounded depth r , where the cost $F(G_Q)$ of a tree G_Q is the sum of its edge weights.
- A class of **Steiner graph-based queries (SG)** [10] that computes r -cliques, where the cost $F(G_Q)$ of an r -clique G_Q is the total pairwise distance among the content nodes.

GExp maintains a built-in library to bookkeep registered KWS query classes, cost models, and query evaluation algorithms. By default, the cost of $Q(G)$ is defined as the sum of the answer cost $F(G_Q)$ for each $G_Q \in Q(G)$. New KWS query classes, algorithms and cost functions can be easily plugged into GExp.

2.2 Cost-aware Query Expansion

We next introduce the query expansion component of GExp. GExp characterizes good query suggestion with a notion of δ -expansion. **δ -expansion.** Given a query Q from a query class Q and answers $Q(G)$, an expanded query Q' of Q is a δ -expansion of Q , if there exists a non-empty set of answers $Q'(G)$, such that (1) all the content nodes in the answer set $Q(G)$ are contained in $Q'(G)$, and (2) $F(Q'(G)) \leq (1 + \delta)F(Q(G))$. By intuition, δ -expansions refer to queries that have non-empty answers, the answers preserve validated content nodes from $Q(G)$ in the last session, and the answers have a bounded answer cost no more than $(1 + \delta)F(Q(G))$.

Given Q and user-defined δ , GExp suggests a set of δ -expansions Q' without enumerating and evaluating KWS queries. It uses a key enabling technique called answer augmentation.

Figure 1: Graph exploration with δ -expansion

Answer augmentation. The general idea is to “reverse engineer” a KWS process. The query expansion first identifies relevant nodes that match a term t within a bounded hop of neighbors of each answer $G_Q \in Q(G)$, and inserts shortest paths that connect these nodes to some nodes in G_Q . This dynamically induces a subgraph G_Q^A “augmented” from G_Q . It then induces a query expansion Q' by adding term t . The correctness follows from that G_Q^A serves as a valid answer for Q' with bounded answer cost.

A general procedure. GExp performs k rounds of local traversal. Each traversal starts from a set of source nodes V_{S_i} in the i th answer $G_{Q_i} \in Q(G)$, and visits up to a bounded hop of their neighbors using a single source shortest path (SSSP) iterator. Specifically,

- (1) For an SSSP at G_{Q_i} and each term t carried by a node v it visits, GExp computes the additional cost if G_{Q_i} is extended to an augmented structure $G_{Q_i}^A$ for a query $Q' = Q \cup \{t\}$, by “appending” shortest paths from the source node V_{S_i} to v .
- (2) When all SSSP are completed, for each new term t , it computes the total additional cost $\hat{w}$ introduced by extending G_{Q_i} to $G_{Q_i}^A$. If $\hat{w}$ is larger than a threshold C_p (set as $\delta * F(Q(G))$ by default), it prunes query $Q \cup \{t\}$. Otherwise, it adds $Q' = Q \cup \{t\}$ as a δ -expansion.

Performance guarantees. GExp specializes the above procedure for DR, ST, and SG, by simply specifying V_{S_i} as (1) the root of G_{Q_i} for DR, and (2) the node set of G_{Q_i} for ST and SG, respectively. For all cases, GExp incurs bounded time costs determined by the size of r -hop neighbors of $Q(G)$ (denoted by $\|Q(G)\|_r$) and k (summarized below). We invite interested readers to consult [1] for details.

KWS query classes	Expansion Time Cost
DR	$O(\ Q(G)\ _r \log \ Q(G)\ _r + \ Q(G)\ _r)$
ST	$O((\ Q(G)\ + k)(\ Q(G)\ _r \log \ Q(G)\ _r + \ Q(G)\ _r))$
SG	$O(k(\ Q(G)\ _r \log \ Q(G)\ _r + \ Q(G)\ _r))$

Example 1: Fig. 1 demonstrates the general idea of cost-aware exploration. The content nodes are marked in bold and the validated answers are bounded by a dashed-rectangle. (1) Given $Q_1(G)$ of a DR query (Fig. 1(a)) with cost 2, and user-defined $\delta=0.5$, the bounded cost to suggest a new term is $2 * (1 + 0.5) = 3$ ($C_p = 1$). A term Emilio can be suggested as a 1 hop neighbor of $Q_1(G)$, adding a cost 1 (given USA as its root). (2) For $Q_2(G)$ as a Steiner tree with cost 0.2 and $\delta=3.5$, the cost bound is $0.2 * (1 + 3.5) = 0.9$. A term Emilio can be suggested to Q , by “appending” a path from UC Berkeley to Emilio to $Q_2(G)$. (3) Similarly, Chamberlain can be suggested to extend $Q_3(G)$ as r -cliques with bounded additional cost 0.6. Larger δ allows “further” terms to be suggested. $\square$

Figure 2: GExp architecture

Diversified query expansion. GExp supports practical exploration scenarios with diversified queries. Given a set of δ -expansions Q' , GExp adopts a bi-criteria diversification function R defined as $R(Q') = (|Q'| - 1)I(Q') + (2\lambda)D(Q')$, where the function $I(Q') = \sum_{Q' \in Q'} I(Q')$ measures the total relevancy of Q' , and the function $D(Q') = \sum_{Q'_i, Q'_j \in Q'} \text{diff}(Q'_i, Q'_j)$ quantifies the pairwise difference $\text{diff}(Q'_i, Q'_j)$ (a metric) between δ -expansions Q'_i and Q'_j in Q' , and λ is a parameter to trade off relevancy with diversity.

The built-in metric library of GExp supports a number of commonly used relevance and dissimilarity measures for KWS and graph search; some are summarized below.

Function	Definition
$I(Q')$: TF-IDF	TF-IDF(Q' , G)
$I(Q')$: Node relevance	Match/PageRank score
$I(Q')$: Surpriseness [13]	$\frac{pr(Q, t')}{\prod_{t_i \in Q} pr(t_i) \cdot p(t')}$
$I(Q')$: Importance	$\text{AvgDeg}(V(Q'), G)$
$\text{diff}(Q'_i, Q'_j)$: Concept Diff. [4]	$1 - \frac{ N(t'_i) \cap N(t'_j) }{ N(t'_i) \cup N(t'_j) }$
$\text{diff}(Q'_i, Q'_j)$: Summary Diff. [14]	$1 - \frac{ V(Q'_i(G)) \cap V(Q'_j(G)) }{ V(Q'_i(G)) \cup V(Q'_j(G)) }$

Diversification. Finding top- n diversified δ -expansions Q' is (not surprisingly) NP-hard. Given functions $I(\cdot)$ and $D(\cdot)$ defined by any of the above metrics, and a pool of queries obtained from answer augmentation, GExp computes diversified Q' with a 2-approximation, by solving a max-sum diversification problem [8].

2.3 Incremental Query Evaluation

Upon receiving a query Q' expanded from Q , the query evaluation component of GExp *incrementally* computes the answer $Q'(G)$ of Q' by updating $Q(G)$ directly, without re-evaluating $Q'(G)$ from scratch. The key enabling technique extends our prior work that copes with changes of *data* [7, 11, 12] to changes of *queries*.

A general procedure. The core idea is to dynamically identify a small set of “edge insertions” ΔE for $Q(G)$, such that $Q'(G)$ can be (approximately) computed directly from $Q(G) \oplus \Delta E$ (where $\oplus$ applies edge insertions to $Q(G)$), via a fast post-processing that only visits $Q(G) \oplus \Delta E$. To find a minimal amount of edges ΔE , GExp performs a backward breadth-first traversal from the content nodes of $Q(G)$ to the nearest nodes that match term t , with an estimated upper bound of answer cost bounded by $\delta F(Q(G))$, for each term $t \in Q' \setminus Q$. ΔE consists of all the edges visited by the traversals.

Example 2: Recall the answer $Q_2(G)$ in Fig. 1(b). Given an expanded query $Q'_2 = Q_2 \cup \text{Emilio}$, GExp computes a Steiner tree by adding to the Steiner tree $Q_2(G)$ with ΔE (marked in red; Fig. 1(d)), induced by the backward traversal from its content nodes. This provides a new answer with cost 0.5, which suggests that all content nodes are scientists and Nobel Prize winners. $\square$

Performance guarantees. Under the practical setting that the content nodes of $Q(G)$ should be preserved by $Q'(G)$, we have developed incremental approximation algorithms to compute $Q'(G)$ with various optimality guarantees, to cope with NP-hard KWS queries (summarized below). The algorithms of all the query classes are special cases of the general procedure, by setting ΔE accordingly. For example, for ST queries, GExp guarantees an 1.5-approximation for $Q'(G)$ when $Q(G)$ is an optimal answer, by setting ΔE as the shortest paths that minimize the distance from some node of G_Q to a content node of t , for each $G_Q \in Q(G)$ and term $t \in Q' \setminus Q$ (see detailed analysis in [1]).

KWS query classes	Incremental evaluation cost	Optimality
DR	$O(\ Q(G)\ _r \log \ Q(G)\ _r + \ Q(G)\ _{2r})$	Optimal
ST	$O(k * (\ Q(G)\ _r \log \ Q(G)\ _r + \ Q(G)\ _r))$	1.5-approx (optimal $Q(G)$)
SG	$O(k \ Q(G)\ _r \log \ Q(G)\ _r + \ Q(G)\ _r)$	Optimal

2.4 GExp Architecture

GExp adopts a three-tier architecture depicted in Fig. 2. (1) The top layer of GExp is an interactive interface that allows users to perform configuration, issue and modify queries, and validate suggested queries and answers. We demonstrate the user-friendly GUI in Section 3. (2) The configuration component maintains built-in libraries of KWS query classes, relevant and dissimilarity functions (for diversified query expansion), and KWS algorithms. Users will be able to register new query classes, semantics, and algorithms via configuration of GExp. (3) At the core of GExp is the cost-aware query expansion component, and incremental query evaluation component. The components invoke answer augmentation, diversified query expansion, and incremental evaluation to construct workflow of GExp, and execute graph exploration in sessions. (4) The storage and index layer maintain underlying graph data and supports fast distance queries and traversal operations.

3 DEMONSTRATION

Setup. We will showcase GExp with real-world large graphs. (1) DBpedia¹, a knowledge graph that contains 4.8M multi-labeled entities, in total 1.5M keywords from both entity names and their type information (e.g., “Place”, “Obama”), and 15M edges with 670 distinct relationships. (2) Citation² is a citation network of 4.3M entities (e.g., papers, authors, publication venues), 16.8M edges including 3 edge types such as cited, published at, and written by. (3) IMDB³ is an information network including 1.6M entities of movies, TV shows, and crews. It contains 5.1M edges and 1.4M tokens from e.g., genre and titles of movies, and the name of crews. The algorithms including the keyword search algorithms for the three query classes [6, 9, 10], query expansion, diversification and relevancy functions were implemented in Java.

¹dbpedia.org

²https://aminer.org/citation

³http://www.imdb.com

Figure 3: GUI: Interactive Query Expansion and Incremental Evaluation

Scenarios. We walk through GExp with the following scenarios.

Ease of use. We invite the users to experience an interactive graph exploratory search with the user-friendly GUI of GExp (Fig. 3). Accessing the “Configuration” panel at any session, users are able to configure the exploration by tuning the exploration range (the size of answers r), quality bound δ , specifying query class and diversification functions. They can monitor and validate the answers to tune the query expansion. The suggested terms, cost estimation, and description of their matches are visualized in “Query Suggestion” panel. The answers are updated upon the selection of new queries and are visualized in “Exploration View” Panel. The cost analysis will be displayed in “Performance” panel.

Diversified exploration. We invite users to experience how GExp supports cost-aware exploration with diversified queries. For example, a DR query $Q_1 = \{Jessica\ Chastain, Anne\ Hathaway\}$ with $\delta = 1$, $r = 1$, and $n = 2$ is posed to IMDB and its top-2 answers are validated by the user. Using TF-IDF as relevance, and concept difference as difference function, GExp suggests top 2 diversified keywords as Taylor Swift and Comedy. In the next session, by selecting Taylor Swift, GExp returns an updated answer with a bounded cost which can also be used for further expansion. By tuning r to 2, a suggested query contains a Spanish TV series titled El hormiguero, which provides the user with new information about actresses who co-played in specific movies with those in Q_1 .

Tuning the exploration. We also demonstrate how a user can tune the parameters δ and r to control the exploration at any time. For example, an ambiguous SG query $Q_2 = \{Knuth, Award, Stanford\}$ with two answers are depicted in Fig. 3. Assuming the user is interested to explore more about answer 2 related to Donald Knuth.

- (1) By setting $\delta = 0.75$ and $r = 2$, GExp finds Sedgewick as a new term with answer cost bounded by 7 and matched by Robert Sedgewick, a Stanford alumni who is Donald’s student.
- (2) By tuning δ to 1 and fixing $r=2$, it finds a second term Rivest with Ron Rivest as a match, a Stanford alumni and also a Turing Award winner, and with answer cost no larger than 8.
- (3) With $\delta = 3$ and $r = 3$, it discovers the expertise of Ron Rivest as Cryptography. These scenarios suggest that GExp can be tuned

to switch between more “conservative” (with lower cost) or more “open” exploration (with more tolerance on quality loss).

Performance of GExp. We visualize the cost analysis for query expansion and incremental evaluation of the sessions. Two charts are shown in Fig. 3, which demonstrate the impact of δ on the cost of query answers, and the response time of sessions, respectively, averaged over 50 queries. We demonstrate that (1) cost-aware exploratory is feasible over large data graphs, (2) the time cost for incremental query evaluation is much less than its counterpart incurred by from-scratch evaluation (KWS algorithms), and illustrate the reasons. We will also compare GExp with ClearMap [3], an interactive XML search engine that focuses on result completion rather than cost-aware exploration. We will demonstrate the unique features of GExp with KWS-based XML data exploration.

4 ACKNOWLEDGMENT

Namaki and Wu are supported in part by NSF IIS-1633629 and Huawei Innovation Research Program (HIRP).

REFERENCES

- [1] Full version. http://eecs.wsu.edu/~mnamaki/GExp_Full.pdf.
- [2] Gexp. <https://github.com/mnamaki/GExp>.
- [3] Z. Bao, Y. Zeng, H. Jagadish, and T. W. Ling. Exploratory keyword search with interactive input. In *SIGMOD*, pages 871–876, 2015.
- [4] A. Bouchoucha, J. He, and J.-Y. Nie. Diversified query expansion using conceptnet. In *CIKM*, pages 1861–1864, 2013.
- [5] C. Carpineto and G. Romano. A survey of automatic query expansion in information retrieval. *CSUR*, 2012.
- [6] B. Ding, J. X. Yu, S. Wang, L. Qin, X. Zhang, and X. Lin. Finding top-k min-cost connected trees in databases. In *ICDE*, pages 836–845, 2007.
- [7] W. Fan, X. Wang, and Y. Wu. Incremental graph pattern matching. *TODS*, 2013.
- [8] S. Gollapudi and A. Sharma. An axiomatic approach for result diversification. In *WWW*, pages 381–390, 2009.
- [9] V. Kacholia, S. Pandit, S. Chakrabarti, S. Sudarshan, R. Desai, and H. Karambelkar. Bidirectional expansion for keyword search on graph databases. In *Vldb*, 2005.
- [10] M. Kargar and A. An. Keyword search in graphs: Finding r -cliques. *Vldb*, 2011.
- [11] M. H. Namaki, P. Lin, and Y. Wu. Event pattern discovery by keywords in graph streams. In *IEEE Big Data*, pages 982–987, 2017.
- [12] M. H. Namaki, K. Sasani, Y. Wu, and T. Ge. Beams: Bounded event detection in graph streams. In *ICDE*, pages 1387–1388, 2017.
- [13] N. Sarkas, N. Bansal, G. Das, and N. Koudas. Measure-driven keyword-query expansion. *PVLDB*, pages 121–132, 2009.
- [14] Q. Song, Y. Wu, and X. L. Dong. Mining summaries for knowledge graph search. In *ICDM*, pages 1215–1220, 2016.
- [15] J. X. Yu, L. Qin, and L. Chang. Keyword search in relational databases: A survey. *IEEE Data Eng. Bull.*, pages 67–78, 2010.