An Empirical Performance Evaluation of Relational Keyword Search Techniques

Joel Coffman, *Member*, *IEEE*, and Alfred C. Weaver, *Fellow*, *IEEE*

Abstract—Extending the keyword search paradigm to relational data has been an active area of research within the database and IR community during the past decade. Many approaches have been proposed, but despite numerous publications, there remains a severe lack of standardization for the evaluation of proposed search techniques. Lack of standardization has resulted in contradictory results from different evaluations, and the numerous discrepancies muddle what advantages are proffered by different approaches. In this paper, we present the most extensive empirical performance evaluation of relational keyword search techniques to appear to date in the literature. Our results indicate that many existing search techniques do not provide acceptable performance for realistic retrieval tasks. In particular, memory consumption precludes many search techniques from scaling beyond small data sets with tens of thousands of vertices. We also explore the relationship between execution time and factors varied in previous evaluations; our analysis indicates that most of these factors have relatively little impact on performance. In summary, our work confirms previous claims regarding the unacceptable performance of these search techniques and underscores the need for standardization in evaluations—standardization exemplified by the IR community.

Index Terms—Keyword search, relational database, information retrieval, empirical evaluation

1 Introduction

The ubiquitous search text box has transformed the way people interact with information. Nearly half of all Internet users use a search engine daily [1], performing in excess of 4 billion searches [2]. The success of keyword search stems from what it does not require—namely, a specialized query language or knowledge of the underlying structure of the data. Internet users increasingly demand keyword search interfaces for accessing information, and it is natural to extend this paradigm to relational data. This extension has been an active area of research throughout the past decade. Despite a significant number of research papers being published in this area, no research prototypes have transitioned from proof-of-concept implementations into deployed systems. The lack of technology transfer coupled with discrepancies among existing evaluations indicates a need for a thorough, independent empirical evaluation of proposed search techniques.

As part of previous work in this area, we created the first benchmark to evaluate relational keyword search techniques [3]. This benchmark satisfies calls [4], [5] from the research community to standardize the evaluation of these search techniques, and our evaluation of search effectiveness [3] revealed that many search techniques perform comparably despite contrary claims in the literature. During our evaluation of search effectiveness, we were surprised by the difficulty we had searching our data sets. In particular, straightforward implementations of many search techniques could not scale to databases with hundreds of thousands of tuples, which forced us to write "lazy" versions of their core algorithms and reduce their memory footprint. Even then, we were surprised by the excessive runtime of many search techniques. Other researchers have recently reported similar experiences. Baid et al. state [6],

> [...] current [keyword search] solutions have unpredictable performance issues. Specifically, while the systems produce answers quickly for many queries, for many others they take an unacceptably long time, or even fail to produce any answer after exhausting memory.

Our shared experience with existing search techniques suggests that the ad hoc evaluations that appear in the literature are inadequate. This sentiment is supported by our survey of existing evaluations [3] and by others who are familiar with the practices established by the IR community for the evaluation of retrieval systems (e.g., see Webber [5]). In this paper, we augment our prior work [3] with an evaluation of existing search techniques' runtime performance. Our findings indicate that much room for improvement exists.

1.1 Overview of Relational Keyword Search

Keyword search on semistructured data (e.g., XML) and relational data differs considerably from traditional IR.[1] A discrepancy exists between the data's physical storage and a logical view of the information. Relational databases are normalized to eliminate redundancy, and foreign keys identify related information. Search queries frequently cross these relationships (e.g., a subset of search terms is

- *J. Coffman is with the Johns Hopkins University Applied Physics Laboratory, 11100 Johns Hopkins Road, Laurel, MD 20723. E-mail: jcoffman@cs.virginia.edu.*
- *A.C. Weaver is with the Department of Computer Science, School of Engineering and Applied Science, University of Virginia, 85 Engineer's Way, PO Box 400740, Charlottesville, VA 22904-4740. E-mail: weaver@cs.virginia.edu.*

Manuscript received 19 Dec. 2011; revised 31 May 2012; accepted 5 Nov. 2012; published online 16 Nov. 2012.
Recommended for acceptance by S. Sudarshan.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-2011-12-0791.
Digital Object Identifier no. 10.1109/TKDE.2012.228.

1. In this paper, we focus on keyword search techniques for relational data. We do not discuss XML retrieval.

TABLE 1
Example of Contradictory Results in the Literature

	execution time (s)					
	DBLP			IMDb		
System	[8]	[9]	[10]	[8]	[9]	[10]
	Evaluation					
BANKS [11]	14.8		5.9	5.0		10.6
BANKS-II [8]	0.7	44.7	7.9	0.6	5.9	6.6
BLINKS [9]		1.2	19.1		0.2	2.8
STAR [10]			1.2			1.6

present in one tuple and the remaining terms are found in related tuples), which forces relational keyword search techniques to recover a logical view of the information. The implicit assumption of keyword search—that is, the search terms are related—complicates the search process because typically there are many possible relationships between search terms. It is frequently possible to include another occurrence of a search term by adding tuples to an existing result. This realization leads to tension between the compactness (and consequently performance) and coverage of search results.

Composing coherent search results from discrete tuples is the primary reason that searching relational data is significantly more complex than searching unstructured text. Unstructured text allows indexing information at the same granularity as the desired results (e.g., by documents or sections within documents). This task is impractical for relational data because an index over logical (or materialized) views is often an order of magnitude larger than the original data [6], [7]. Such an approach will not scale to large databases such as those underlying electronic medical records (EMRs) or social networking sites.

1.2 Motivation

As we discuss later in this paper, many relational keyword search techniques approximate solutions to intractable problems. Although worst case performance bounds for many of these algorithms have been established, they perform much better in practice than their algorithmic analysis might suggest. Researchers consequently use empirical evaluation to ascertain the benefits of proposed search techniques. Another motivation for this work is the discrepancies among existing evaluations that litter the literature. Table 1 lists the mean execution times of search techniques from three evaluations that use data sets from DBLP[2] and the Internet Movie Database (IMDb)[3]. The table rows are search techniques; the columns are different evaluations of these search techniques. Empty cells indicate that the technique was not included in that evaluation. The table illustrates two concerns that we have regarding existing evaluations.

First, the difference in the relative runtime performance of each search technique is startling. We do not expect the most recent evaluation to downgrade the *orders of magnitude* performance improvements to performance degradations, which is certainly the case on the DBLP data set. Second, the absolute execution times for the search techniques vary widely across different evaluations. The original evaluation of each approach claims to provide "interactive" response times (on the order of a few seconds), but the other evaluations presented in Table 1 strongly refute this claim. Hence, there remains considerable uncertainty regarding both the relative and absolute performance of existing search techniques. Both of these concerns warrant independent evaluation to establish a performance baseline for realistic retrieval tasks.

2. http://dblp.uni-trier.de/.
3. http://www.imdb.com/.

1.3 Contributions and Outline

In previous work [3], we proposed the first benchmark to evaluate relational keyword search techniques and evaluated them with regard to their search effectiveness. However, our previous work did not consider the runtime performance of these search techniques, which is our focus in this paper. Unlike many evaluations that appear in the literature, our benchmark uses realistic data sets and realistic queries to investigate the numerous tradeoffs made in the design of these search techniques. Our benchmark is the only one to date in the literature that satisfies the *minimum* criteria established by the IR community for the evaluation of retrieval systems.

The major contributions of this paper are as follows:

- We conduct an independent, empirical evaluation of the runtime performance of seven relational keyword search techniques. Our evaluation is the most extensive and thorough one to appear to date in the literature.
- Our results do not substantiate previous claims regarding the scalability and performance of relational keyword search techniques. Existing search techniques perform poorly on databases exceeding tens of thousands of tuples or require an inordinate amount of memory.
- We show that many parameters varied in existing evaluations are at best loosely correlated with runtime performance. The lack of a meaningful relationship gives merit to previous claims of unpredictable performance [6] for existing search techniques.
- Our work is the first to combine performance and search effectiveness in the evaluation of such a large number of search techniques. Considering these two issues in conjunction provides better understanding of these two critical tradeoffs among competing approaches.

The remainder of this paper is organized as follows: Section 2 formally defines the problem of keyword search in relational data graphs and describes the search techniques included in our evaluation. Section 3 describes our experimental setup, including our evaluation benchmark and metrics. In Section 4, we present our experimental results, and we discuss them in Section 5. We review related work in Section 6 and provide our conclusions in Section 7. Online appendices provide greater detail about our evaluation benchmark and summarize implementation details of the search techniques.

TABLE 2
Algorithmic Worst Case Analysis of Graph-Based Search Techniques

System	Performance Ratio	Time	Memory
BANKS [11]	$O(\|Q\|)$	$O(\|V\|^2 \log \|V\| + \|V\| \cdot \|E\|)$	$O(\|Q\| \cdot \|V\|^2)$
BANKS-II [8]	$O(\|Q\|)$	$O(\|V\|^2 \log \|V\| + \|V\| \cdot \|E\|)$	$O(\|Q\| \cdot \|V\|)$
DPBF [12]	1	$O(3^{\|Q\|} \cdot \|V\| + 2^{\|Q\|} \cdot ((\|Q\| + \log \|V\|) \cdot \|V\| + \|E\|))$	$O(2^{\|Q\|} \cdot \|V\|)$
BLINKS [9]	$O(\|Q\|)$	$\langle$dependent on partitioning$\rangle$	$O(\sum_b N_b^2 + BP)$
STAR [10]	$O(\log \|Q\|)$	$O(\frac{1}{\epsilon} \cdot \frac{\max_e weight(e)}{\min_e weight(e)} \cdot \|E\| \cdot \|T\| \cdot (\|V\| \log \|V\| + \|E\|))$	$O(\|Q\| \cdot \|V\|)$

$|V|$: number of nodes (tuples) in data graph. $|E|$: number of edges (foreign keys) in data graph. $|T|$: number of unique terms in database. $|Q|$: number of terms in query. N_b: size of block b in index. B: number of blocks in index. P: number of node separators in node-based partitioning of graph.

2 Relational Search Techniques

Given our focus on empirical evaluation, we adopt a general definition of keyword search over data graphs. This section also presents the search techniques included in our evaluation.

Problem statement. A relational database is a graph $G = (V, E)$. Each vertex $v \in V$ corresponds to a tuple in the relational database. An edge $(u, v) \in E$ represents each relationship (i.e., foreign key) in the relational database. Each vertex is decorated with the set of terms it contains. A query Q is a set of terms. A result for Q is a tree T that is reduced with respect to $Q' \subseteq Q$; that is, T contains all the terms of Q' but no proper subtree that also contains all of them.[4] Results are ranked in decreasing order of their estimated relevance to the information need expressed by Q.

Schema-based approaches support keyword search over relational databases via direct execution of SQL commands. The database's full text indexes identify all tuples that contain search terms, and a join expression is created for each possible relationship between these tuples. DISCOVER [16] pioneered this general approach and ranks results by the number of joins in the SQL query. Hristidis et al. [17] later refined DISCOVER by adopting pivoted normalization weighting [18] to rank results. Top-k query processing strategies provide efficient execution.

The objective of graph-based approaches is to minimize the weight of result trees. This task is a formulation of the group Steiner tree problem [19], which is known to be NP-complete [20]. BANKS [11] enumerates results by searching the graph backwards from vertices that contain query keywords. BANKS-II [8] also searches the graph forwards from potential root nodes of results. DPBF [12] is a dynamic programming algorithm to find the optimal group Steiner tree but remains exponential in the number of search terms. He et al. [9] propose a bi-level index to improve the performance of bidirectional search [8]. STAR [10] is a pseudopolynomial-time algorithm for the Steiner tree problem. It computes an initial solution quickly and then improves this result iteratively.

Table 2 compares the graph-based approaches by worst case execution time and memory requirements. The schema-based approaches are not included in the table due to their loose algorithmic upper bounds. A search technique's performance ratio is its approximation bound on its ideal answer (i.e., computing the optimal group Steiner tree). As evidenced by the table, the worst case execution times and memory consumption vary widely, and these upper bounds are unlikely to be realized in practice. Thus, while algorithmic analysis has been used in the literature to argue for the superiority of particular search techniques, the lack of tight (lower) bounds dictates that these approaches be evaluated empirically.

4. Alternative semantics are also possible (e.g., defining a result as a graph [13], [14], [15]), and some systems described in this section deviate slightly from this definition (e.g., the root of a result tree should have more than 1 child [8], [11]).

3 Evaluation Framework

In this section, we present our evaluation framework. We start with our benchmark and then describe our metrics and experimental setup. We refer the reader to the benchmark's original description [3] for additional details that space precludes us from repeating here.

3.1 Benchmark Overview

Our evaluation benchmark includes the three data sets shown in Table 3: MONDIAL [21], IMDb, and Wikipedia. Two data sets (IMDb and Wikipedia) are extracted from popular websites. As shown in Table 3, the size of the data sets varies widely: MONDIAL is more than two orders of magnitude smaller than the IMDb data set, and Wikipedia lies in between. In addition, the schemas and content also differ considerably. MONDIAL has a complex schema with almost 30 relations while the IMDb subset has only 6. Wikipedia also has few relations, but it contains the full text of articles, which emphasizes sophisticated ranking schemes for results. Our data sets roughly span the range of data set sizes that have been used in other evaluations even though our IMDb and Wikipedia data sets are both subsets of original databases. Using a database subset likely overstates the efficiency and effectiveness of evaluated search techniques.

The benchmark's query workload is derived from 50 information needs for each data set. The query workload

TABLE 3
Characteristics of the Evaluation Data Sets

			in *thousands*		
Dataset	Size (MBs)	Relations	$\|V\|$	$\|E\|$	$\|T\|$
MONDIAL	16	28	17	56	12
IMDb	459	6	1673	6075	1748
Wikipedia	391	6	206	785	750

$|V|$: number of nodes (tuples) in data graph. $|E|$: number of edges (foreign keys) in data graph. $|T|$: number of unique terms.

TABLE 4
Query and Result Statistics

	Search log [22]	Benchmark			Results	
Dataset	$\overline{[\![q]\!]}$	$\lvert Q \rvert$	$[\![q]\!]$	$\overline{[\![q]\!]}$	$[\![R]\!]$	$\overline{[\![R]\!]}$
MONDIAL		50	1–5	2.04	1–35	5.90
IMDb	2.71	50	1–26	3.88	1–35	4.32
Wikipedia	2.87	50	1–6	2.66	1–13	3.26
Overall	2.37	150	1–26	2.86	1–35	4.49

$\lvert Q \rvert$: total number of queries. $[\![q]\!]$: range in number of query terms. $\overline{[\![q]\!]}$: mean number of terms per query. $[\![R]\!]$: range in number of relevant results per query. $\overline{[\![R]\!]}$: mean number of relevant results per query.

does not use real user queries extracted from a search engine log for two reasons. First, Internet search engine logs do not contain queries for data sets not derived from websites. Second, many queries are inherently ambiguous and knowing the user's original information need is essential for accurate relevance assessments. Consequently, we independently derived a variety of information needs for each data set.

The gold standard for relevance judgments was obtained by constructing SQL queries that retrieved all possible relevant results for each information need. The results returned by the SQL queries were manually judged for relevance where—in keeping with the definition of relevance established by the IR community—relevant results must address the query's information need, not just contain all search terms.

Table 4 provides the statistics of the query workload and relevant results for each data set. Five IMDb queries are outliers because they include an exact quote from a movie. Omitting these queries reduces the maximum number of terms in any query to 7 and the mean number of terms per query to 2.91. In-depth analysis [23] indicates that our query workload is far more consistent with real user queries than query workloads used in previous evaluations.

3.2 Metrics

We use two metrics to measure runtime performance. The first is execution time, which is the time elapsed from issuing a query until an algorithm terminates. Because there are a large number of potential results for each query, search techniques typically return only the top-k results where k specifies the desired retrieval depth. Our second metric is response time, which we define as the time elapsed from issuing the query until i results have been returned (where $i \leq k$). Because this definition is not well-defined when fewer than k results are retrieved, we define it for j, where $i < j \leq k$ and i is the number of results retrieved and k is the desired retrieval depth, as the algorithm's execution time.

Effectiveness metrics are also critical to the evaluation of retrieval systems because not every result is actually relevant to the query's underlying information need. Recall is the ratio of relevant results retrieved to the total number of relevant results. Precision is the ratio of relevant results retrieved to the total number of retrieved results. Precision @ k (P@k) is the mean precision across multiple queries where the retrieval depth is limited to k results. If fewer than k results are retrieved by a system, we calculate the precision value at the last result. We also use MAP to measure retrieval effectiveness at greater retrieval depths.

Measuring the completeness of the set of of search results returned by a particular search technique is tempting, but only Golenberg et al.'s algorithm [24] is proven to be complete (i.e., return all possible results) for the given search terms. Furthermore, it is not clear what effect omitting some results may have on a search technique. Unlike recall, which is measured against the set of *relevant* results, omitting a few results may have practically no impact on the effectiveness of the search technique, particularly if the omitted results are highly redundant with others that are enumerated [24]. More importantly, there is no precedent from the IR community to evaluate retrieval systems using a purely objective metric because retrieval systems explicitly answer subjective information needs.

3.3 Implementations

We reimplemented BANKS, DISCOVER, DISCOVER-II, and DPBF and obtained implementations of BANKS-II (i.e., the bidirectional search algorithm), BLINKS, and STAR. All the search techniques are implemented in Java. For some search techniques, we also had access to others' implementations. Among the implementations, we found that our reimplementations generally outperform the implementations provided by others. Exceptions to this trend were the result of correcting significant implementation defects. Our experiments do not compare against traditional IR systems (e.g., Apache Lucene[5]) because more traditional systems do not consider the relationships among database tuples, which is an important aspect of relational keyword search.

Our implementation of BANKS adheres to its original description although it queries the database dynamically to identify nodes (tuples) that contain query keywords. Our implementation of DISCOVER borrows its successor's query processing techniques. Both DISCOVER and DISCOVER-II are executed with the sparse algorithm, which provides the best performance for queries with AND semantics [17]. BLINKS's block index was created using breadth-first partitioning and contains 50 nodes per block.[6] STAR uses the edge weighting scheme proposed by Ding et al. [12] for undirected graphs.

3.4 Experimental Setup

Our experimental setup is comparable to those reported in previous evaluations. We execute each query on a Linux machine running Ubuntu 10.04 with dual 1.6-GHz AMD Opteron 242 processors and 3 GB of RAM. We compiled each implementation using `javac` version 1.6 and ran the implementations with the Java HotSpot 64-bit server VM. PostgreSQL was our database management system.

We impose a maximum execution time of 1 hour for each search technique. If the algorithm has not terminated after this time limit, we stop its execution and denote it as a timeout exception. We allow implementations to use $\approx$5 GB of virtual memory[7] and limit the size of results to

5. http://lucene.apache.org/.
6. Memory consumption scales with the number of blocks.
7. This was the maximum amount we could consistently allocate on our machines without triggering Linux's out-of-memory killer. We also specified `-Xincgc` to enable Java's incremental garbage collector, which was essential for reasonable performance.

TABLE 5
Summaries of Queries Completed and Exceptions

(a) MONDIAL

System	✔	✘ TO	VM	?	exec.
BANKS	29	21	—	—	1910.9
DISCOVER	50	—	—	—	8.0
DISCOVER-II	50	—	—	—	6.5
BANKS-II	50	—	—	—	190.2
DPBF	50	—	—	—	11.1
BLINKS	50	—	—	—	23.6
STAR	50	—	—	—	0.3

(b) IMDb

System	✔	✘ TO	VM	?	exec.
BANKS	7	39	—	4	3239.7
DISCOVER	50	—	—	—	227.9
DISCOVER-II	50	—	—	—	201.8
BANKS-II	—	18	—	32	3604.3
DPBF	5	45	—	—	3399.3
BLINKS	—	—	50	—	—
STAR	—	—	50	—	—

(c) Wikipedia

System	✔	✘ TO	VM	?	exec.
BANKS	11	39	—	—	2966.4
DISCOVER	50	—	—	—	43.1
DISCOVER-II	50	—	—	—	39.8
BANKS-II	13	35	—	2	2912.7
DPBF	47	3	—	—	732.1
BLINKS	—	—	50	—	—
STAR	3	—	47	—	22.4

√: total queries completed successfully (out of 50), more are better. ×: query failures, fewer are better. TO: timeout exceptions (> 1 hour of execution time). VM: memory exceptions (exhausted virtual memory). ?: either a timeout or memory exception (see accompanying text). exec.: mean execution time (in seconds) across all queries, lower is better.

five nodes (tuples). Once a search technique consumes the available physical memory, the operating system's virtual memory manager is responsible for paging data to and from disk. If an algorithm exhausts the total amount of heap memory, we mark it as failing due to excessive memory requirements. All values reported in our experiments are the mean of three different executions of each search technique.

4 EXPERIMENTS

Table 5 lists the number of queries executed successfully by each search technique for our data sets and also the number and types of exceptions we encountered. Of interest is the number of queries that were not completed successfully. Queries fail due to time outs (i.e., the algorithm had not terminated after 1 hour of execution time) or exhausting virtual memory. In the table, these exceptions are indicated by "TO" and "VM." Unfortunately, the cause of a search technique's failure is not always apparent, particularly when the system is thrashing due to the use of virtual memory. Severe thrashing prevents graceful cleanup when the time limit expires because it can take a considerable amount of time to page in the error handling code—longer than the 15 minutes that we permitted before the scheduler killed the job. Likewise, correctly identifying the exhaustion of heap space is challenging because it can be difficult to handle Java's OutOfMemoryError.[8] When the garbage collector cannot free any memory, it may not be possible to execute even our minimal error handling code. Hence, the root cause for some failures (i.e., timeout or memory exception) remains unknown and is indicated in the table by "?".[9]

Most search techniques complete all the MONDIAL queries with mean execution times ranging from less than a second to several hundred seconds. Results for IMDb and Wikipedia are more troubling. Only DISCOVER and DISCOVER-II complete all the IMDb queries, and their mean execution time is several minutes. DPBF comes close to completing the Wikipedia queries but still has several timeout exceptions, and both DISCOVER and DISCOVER-II require in excess of half a minute on average to complete these queries.

To summarize these initial results, existing search techniques provide reasonable performance only on the smallest data set (MONDIAL). Performance degrades significantly when we consider a data set with hundreds of thousands of tuples (Wikipedia) and becomes unacceptable for a data set with a million tuples (IMDb). The memory consumption for the graph-based approaches is considerable, which prevents most search techniques from completing the IMDb queries.

4.1 Execution Time

Fig. 1 shows box plots of the execution times for all queries on each data set. The box plots confirm the performance trends in Table 5 but also illustrate the variation in execution time among different queries. In particular, the range in execution times for a search technique is often several orders of magnitude. Most search techniques also have outliers in their execution times; these outliers indicate that the performance of these search heuristics varies considerably. Previous evaluations—most of which report only the mean execution time for queries—have not acknowledged the existence of such outliers.

4.1.1 Number of Search Terms

A number of previous evaluations [8], [12], [16], [17] report mean execution time for queries that contain different numbers of search terms to show that performance remains acceptable even when queries contain more keywords. Fig. 2 graphs these values for the different search techniques. Some search techniques fail to complete some queries, which accounts for the omissions in the graph. As evidenced by the graph, queries that contain more search terms require more time to execute on average than queries that contain fewer search terms. The relative performance

8. The Java documentation states that "a reasonable application should not try to catch [a virtual machine error]."

9. Reducing the amount of virtual memory (e.g., to match the machine's physical memory) would prevent this uncertainty, but some search techniques cannot even search the MONDIAL database with less memory [25].

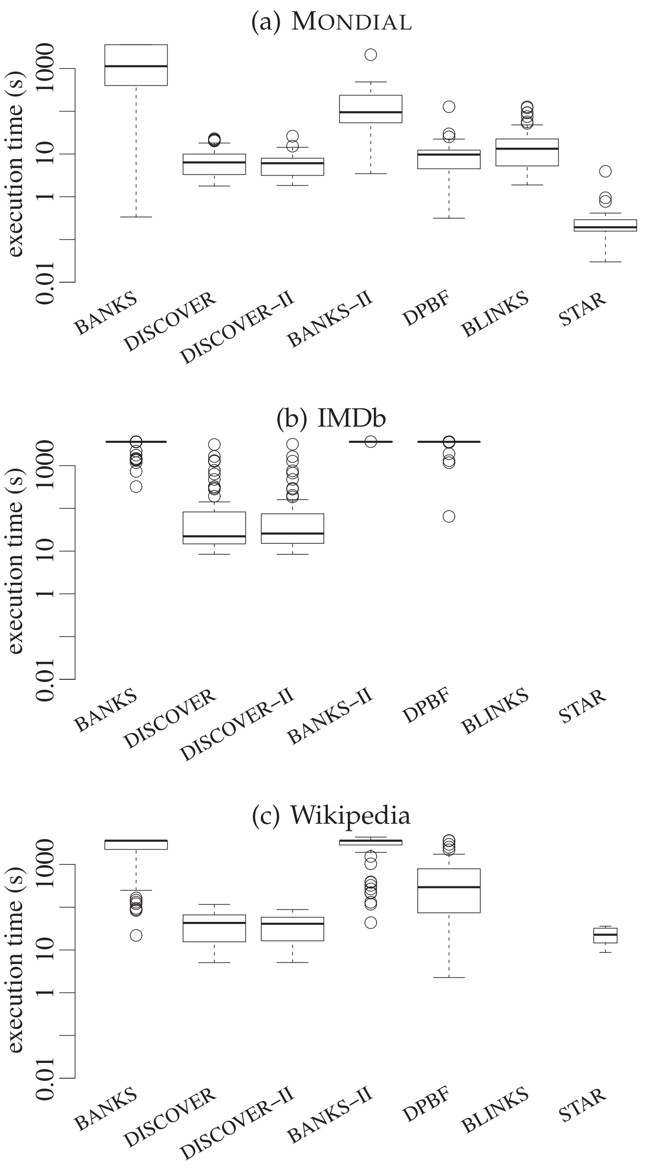


Fig. 1. Box plots of the execution times of each search technique (lower is better). Note that the y-axis has a log scale. Search techniques are ordered by publication date and the retrieval depth was 100 results.

among the different search techniques is unchanged from Fig. 1 although we do see that DPBF outperforms the schema-based approaches on queries with only a single term. DPBF's performance falters with additional search terms, which is consistent with its algorithmic analysis—exponential in the number of query terms.

These results are similar to those published in previous evaluations, but using Fig. 2 as evidence for the efficiency of a particular search technique can be misleading. In Fig. 3, we show box plots of the execution times of BANKS and DISCOVER-II for MONDIAL queries to illustrate the range in execution times. As evidenced by these graphs, several queries have execution times much higher than the rest. These queries give the search techniques the appearance of unpredictable performance, particularly when the query is similar to another one that completes quickly.

For example, the query "Uzbek Asia" for BANKS has an execution time three times greater than the query "Hutu Africa." DISCOVER-II has similar outliers; the query "Panama Oman" requires 3.5 seconds to complete even though the query "Libya Australia" completes in less than half that time. From a user's perspective, these queries would be expected to have similar execution times. These outliers (which are even more pronounced for the other data sets) suggest that simply looking at mean execution time for different numbers of query keywords does not reveal the complete performance profile of these systems. Moreover, existing work does not adequately explain the existence of these outliers and how to improve the performance of these queries.

4.1.2 *Collection Frequency*

In an effort to better understand another factor that is commonly cited as having a performance impact, we consider the frequency of search terms in the database (see Fig. 4). We do not show IMDb and Wikipedia because they are comparable to the MONDIAL results albeit significantly higher mean execution times. The results are surprising: execution time appears relatively uncorrelated with the number of tuples containing search terms.

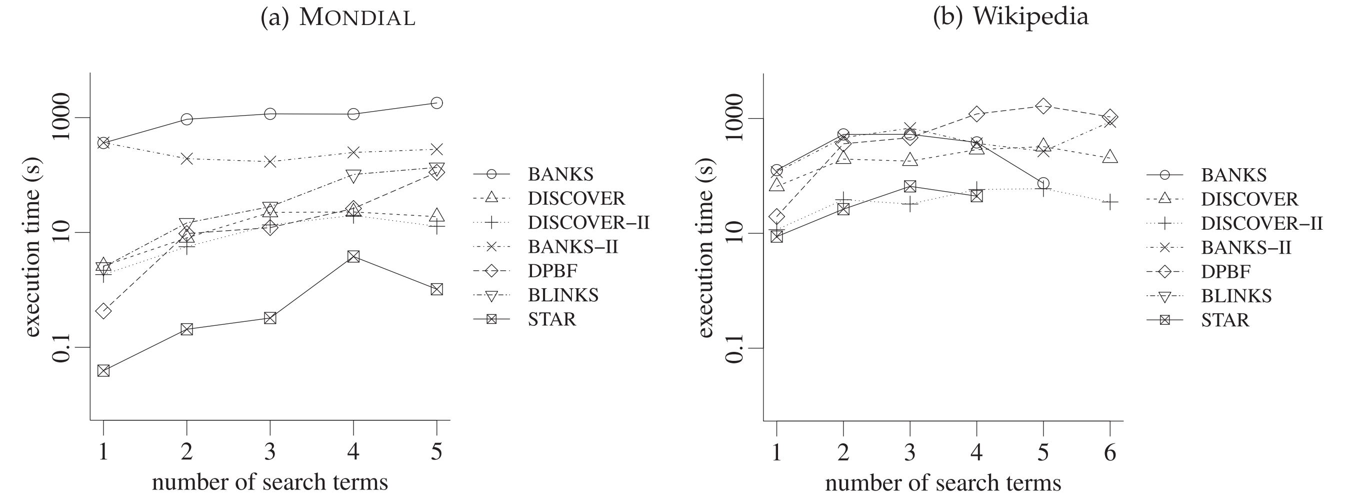


Fig. 2. Execution time versus query length for MONDIAL and Wikipedia queries; lower execution times are better. IMDb is omitted due to the few search techniques that successfully complete its queries. Note that the y-axis has a log scale. The retrieval depth was 100 results.

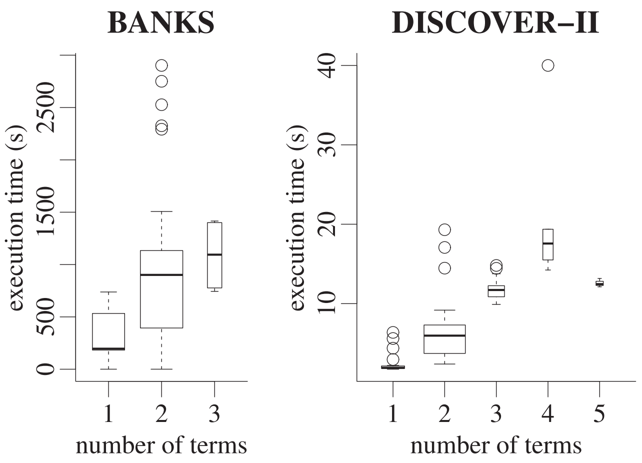


Fig. 3. Box plots of execution times for BANKS (left) and DISCOVER-II (right) on MONDIAL with a retrieval depth of 100. The width of the box reflects the number of queries in each sample.

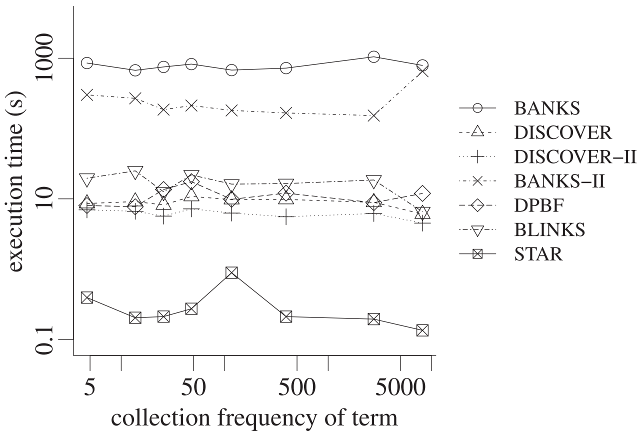


Fig. 4. Execution time versus frequency of query terms in the MONDIAL database. Lower execution times are better. Note that the x-axis and y-axis have a log scale. The retrieval depth was 100 results.

This result is unexpected, for one expects the time to increase when more nodes (and all their relationships) must be considered. One possible explanation for this phenomenon is that the search space in the interior of the data graph (i.e., the number of nodes that must be explored when searching) is not correlated with the frequency of the keywords in the database. He et al. [9] imply the opposite; we believe additional experiments are warranted as part of future work to determine what effect the underlying data graph has on this result.

4.1.3 Retrieval Depth

Table 6 considers the scalability of the various search techniques at different retrieval depths—10 and 100 results. Overall, we see considerable variation in mean execution time at different retrieval depths. Retrieval depth has practically no impact for DISCOVER and DISCOVER-II whereas it is significant for the other search techniques. In general, the percentage slowdown for each search technique is similar across the data sets. While it would be interesting to extend this analysis to greater retrieval depths (e.g., 1,000 results), the failure for most search techniques to complete the IMDb and Wikipedia queries undermines the value of such experiments.[10]

TABLE 6
Performance Comparison at Different Retrieval Depths

(a) MONDIAL

	execution time (s)		slowdown	
System	$k = 10$	$k = 100$	Δ	%
BANKS	1617.5	1910.9	293.4	18.1
DISCOVER	6.8	8.0	1.2	17.6
DISCOVER-II	7.1	6.5	-0.6	-8.5
BANKS-II	79.1	190.2	111.1	140.5
DPBF	5.4	11.1	5.7	105.6
BLINKS	15.2	23.6	8.4	55.3
STAR	0.2	0.3	0.1	50.0

(b) IMDb

	execution time (s)		slowdown	
System	$k = 10$	$k = 100$	Δ	%
BANKS	2329.2	3239.7	910.5	39.1
DISCOVER	178.1	227.9	49.7	27.9
DISCOVER-II	178.1	201.8	23.6	13.2
BANKS-II	3604.1	3604.3	0.2	0.0
DPBF	2012.3	3399.3	1387.0	68.9
BLINKS	—	—	—	—
STAR	—	—	—	—

(c) Wikipedia

	execution time (s)		slowdown	
System	$k = 10$	$k = 100$	Δ	%
BANKS	1723.3	2966.4	1243.1	72.1
DISCOVER	40.9	43.1	2.2	5.4
DISCOVER-II	39.7	39.8	0.1	0.3
BANKS-II	877.4	2912.7	2035.3	232.0
DPBF	280.8	732.1	451.3	160.7
BLINKS	—	—	—	—
STAR	86.8	22.4	-64.4	-74.2

k: retrieval depth. Δ: absolute change in mean execution time, lower is better. %: percentage change in mean execution time, lower is better.

4.1.4 Result Size

Table 7 provides the number of successful queries (i.e., queries that did not time out or exhaust virtual memory) and mean execution time for our data sets when we vary the maximum size of result trees (i.e., number of tuples from the underlying database)[11] BLINKS and STAR are not reported in this experiment because neither provides a parameter to limit the search space to trees of a maximum size. The table reveals several interesting trends.

First, as result size increases, fewer queries are completed successfully by the search techniques. This result is expected because the number of possible results grows exponentially with the size of allowable results. Second, mean execution time generally increases when the size of the search space increases. The notable exception to these two trends is BANKS whose number of exceptions and

10. MONDIAL's small size is not amenable to specifying larger retrieval depths because most search techniques already identify all the relevant results. A larger retrieval depth would merely measure how quickly the different search techniques exhaust the possible search space.

11. The results in this table use a single experimental run of each search technique.

TABLE 7
Queries Completed and Execution Time for Different Result Sizes

(a) MONDIAL

size	3		7		9	
System	✓	exec.	✓	exec.	✓	exec.
BANKS	28	1968.1	32	1708.1	39	1411.4
DISCOVER	50	2.2	40	820.6	15	2489.1
DISCOVER-II	50	2.3	40	837.3	15	2535.6
BANKS-II	50	53.8	50	229.8	50	391.7
DPBF	50	2.5	50	22.2	50	31.0

(b) IMDb

size	3		7		9	
System	✓	exec.	✓	exec.	✓	exec.
BANKS	1	3529.8	10	3108.8	10	3099.6
DISCOVER	50	12.7	40	963.8	31	1773.1
DISCOVER-II	50	112.4	40	1122.1	24	2274.4
BANKS-II	0	—	0	—	0	—
DPBF	24	2176.7	5	3406.6	5	3398.9

(c) Wikipedia

size	3		7		9	
System	✓	exec.	✓	exec.	✓	exec.
BANKS	18	2977.4	15	2614.1	26	2350.9
DISCOVER	50	46.1	49	308.3	40	1124.8
DISCOVER-II	50	40.2	50	306.7	37	1779.2
BANKS-II	46	515.6	11	2912.7	8	3035.5
DPBF	50	184.7	38	1274.4	38	1358.0

$\surd$: Queries completed successfully (out of 50), higher is better. exec.: mean execution time (in seconds) across all queries, lower is better.

TABLE 8
Mean Response Time to Retrieve the Top-k Query Results

(a) MONDIAL

		$k = 1$		$k = 10$	
System	exec.	resp.	%	resp.	%
BANKS	1910.9	1681.8	88.0	1778.9	93.1
DISCOVER	8.0	8.0	100.0	8.0	100.0
DISCOVER-II	6.5	6.5	100.0	6.5	100.0
BANKS-II	190.2	54.2	28.5	143.4	75.4
DPBF	11.1	1.9	17.1	5.2	46.8
BLINKS	23.6	8.4	35.6	14.1	59.7
STAR	0.3	0.3	100.0	0.3	100.0

(b) IMDb

		$k = 1$		$k = 10$	
System	exec.	resp.	%	resp.	%
BANKS	3239.7	2614.7	80.7	2686.0	82.9
DISCOVER	227.9	227.9	100.0	227.9	100.0
DISCOVER-II	201.8	201.8	100.0	201.8	100.0
BANKS-II	3604.3	3604.3	100.0	3604.3	100.0
DPBF	3399.3	1371.8	40.4	2042.1	60.1
BLINKS	—	—	—	—	—
STAR	—	—	—	—	—

(c) Wikipedia

		$k = 1$		$k = 10$	
System	exec.	resp.	%	resp.	%
BANKS	2966.4	2073.7	69.9	2229.9	75.2
DISCOVER	43.1	43.1	100.0	43.1	100.0
DISCOVER-II	39.8	39.8	100.0	39.8	100.0
BANKS-II	2912.7	2635.3	90.5	2755.8	94.6
DPBF	732.1	67.5	9.2	251.3	34.3
BLINKS	—	—	—	—	—
STAR	22.4	22.4	100.0	22.4	100.0

resp.: mean response time (in seconds), lower is better. exec.: mean total execution time (in seconds) to retrieve 100 results, lower is better. %: percentage of total execution time, lower is better.

mean execution time actually decrease when larger results are allowed. The reason for this behavior is that BANKS's enumeration algorithm returns results in approximate order. These results are buffered in a fixed-size heap, and only after the heap is full (or the search space is exhausted) will BANKS start to output results. Hence, expanding the search space allows BANKS to identify more results than it can when the search space is restricted, and finding more results lowers its total execution time.

The final trend we want to highlight is the scalability of the schema-based approaches as result size increases. While the schema-based approaches continue to outperform the graph-based search techniques on IMDb and Wikipedia, they falter on MONDIAL. Despite MONDIAL's small size, it has the most complex schema. DISCOVER and DISCOVER-II's candidate network generation algorithm must enumerate all possible join expressions that contain up to the maximum number of tuples; the total number of join expressions grows exponentially with result size. Hence, MONDIAL's more complex schema does not allow these approaches to scale as well as the graph-based approaches when the maximum result size is increased. This finding is disappointing, particularly because MONDIAL's 30-relation schema is still far simpler than enterprise databases, which can contain hundreds of interrelated tables [26].

4.2 Response Time

In addition to overall search time, the response time of a keyword search system is of critical importance. Systems that support top-k query processing need not enumerate all possible results before outputting some to the user. Outputting a small number of results (e.g., 10) allows the user to examine the initial results and to refine the query if these results are not satisfactory.

Table 8 provides the mean response time to retrieve the first and tenth query result. Interestingly, the response time for most systems is very close to the total execution time, particularly for $k = 10$. The ratio of response time to the total execution time provided in the table shows that some scoring functions are not good at incrementally identifying the best search results. For example, DISCOVER-II identifies the highest ranked search result at the same time as it identifies the tenth ranked result because its bound on the possible score of unseen results falls very rapidly after enumerating more than k results. Although more incremental algorithms exist to enumerate search results, we found that these algorithms did not perform as well as the sparse algorithm we used for our experiments. The notable exception to this general trend is DPBF, which identifies results the most incrementally of any search technique. In general, the graph-based approaches identify results more incrementally than the schema-based approaches.

Another issue of interest is the overhead required to retrieve additional search results. In other words, how much

TABLE 9
Comparison of Total Execution Time and Response Time

(a) MONDIAL

			slowdown	
System	exec.	resp.	Δ	%
BANKS	1617.5	1778.9	161.4	10.0
DISCOVER	6.8	8.0	1.2	17.6
DISCOVER-II	7.1	6.5	-0.6	-8.6
BANKS-II	79.1	143.4	64.3	81.3
DPBF	5.4	5.2	-0.2	-3.7
BLINKS	15.4	14.1	-1.3	-8.4
STAR	0.2	0.3	0.1	50.0

(b) IMDb

			slowdown	
System	exec.	resp.	Δ	%
BANKS	2329.2	2686.0	356.8	15.3
DISCOVER	178.2	227.9	49.7	27.9
DISCOVER-II	178.2	201.8	23.6	13.2
BANKS-II	3604.1	3604.3	0.2	0.0
DPBF	2012.3	2042.1	29.8	1.5
BLINKS	—	—	—	—
STAR	—	—	—	—

(c) Wikipedia

			slowdown	
System	exec.	resp.	Δ	%
BANKS	1723.3	2229.9	507.8	29.5
DISCOVER	40.9	43.1	10.9	5.4
DISCOVER-II	39.7	39.8	0.1	0.3
BANKS-II	877.4	2755.8	1878.4	214.1
DPBF	280.8	251.3	-29.5	-10.5
BLINKS	—	—	—	—
STAR	86.8	22.4	-64.4	-74.2

exec.: total execution time (in seconds) when retrieving 10 results. resp.: response time (in seconds) to retrieve top-10 of 100 results. △: mean absolute difference in time, lower is better. %: mean percentage difference in time, lower is better.

additional time is spent maintaining enough state to retrieve 100 results instead of just 10? Table 9 gives the execution time to retrieve 10 results and the response time to retrieve the first 10 results of 100. The percentage slowdown reveals that the overhead is minimal for most of the search techniques. This result is ideal because it does allow the search techniques to scale gracefully to larger retrieval depths. However, we do note that both the evaluation and response times remain noninteractive (i.e., requiring more than a few seconds) on all but the MONDIAL database.

4.3 Memory Consumption

We measured each search technique's memory footprint immediately prior to executing a query to better understand the memory utilization of the systems. This analysis used a machine with 32 GB of memory to investigate the impact that limited memory has on our experiments. The results are shown in Table 10.

As evidenced by the table, the schema-based systems consume very little memory, most of which is used for the database schema. In contrast, the graph-based approaches require considerably more memory to store their data graph. BLINKS's total memory consumption is particularly high due to its bi-level index, which scales quadratically with the number of nodes per block and is more than an order of magnitude larger than the original data graph. Even with 32 GB of memory, BLINKS cannot complete the construction of its index for the IMDb and Wikipedia data sets. Worse, we found that less than 20 percent of the index had been created for these data sets, which suggests that BLINKS would need in excess of 160 GB of memory for these database subsets! STAR is unique in that the implementation we obtained [10] searches from terminals—i.e., the data graph includes nodes for each search term and edges from each node to the terms that node contains. Essentially, STAR solves the Steiner problem instead of the group Steiner problem. This difference accounts for STAR's greater memory consumption for its data graph. Although we note that STAR can complete more queries for the IMDb and Wikipedia data sets with additional memory,[12] its memory consumption begs the question of just how much memory is reasonable. A representation that is at least an order of magnitude larger than the original relational data not only seems excessive but also will not allow search techniques to scale to the original, complete databases.[13]

TABLE 10
Initial Memory Consumption

	MONDIAL		IMDb		Wikipedia	
System	G	Σ	G	Σ	G	Σ
BANKS	14.1	14.2	1469.5	1469.6	182.2	182.3
DISCOVER	0.2	0.4	0.1	0.2	0.1	0.2
DISCOER-II	0.2	0.4	0.1	0.2	0.1	0.2
BANKS-II	27.4	33.8	3576.0	4641.1	1083.0	1610.5
DPBF	12.5	12.7	1315.4	1315.5	163.0	163.2
BLINKS	28.5	1376.6	—	—	—	—
STAR	75.8	87.1	12907.0	14846.2	4438.5	5350.0

All values are in MBs. G: size of in-memory representation of database. $\sum$: initial memory required by search technique.

4.4 Effectiveness

We do not explore the effectiveness of the various search techniques in detail here due to the extensive coverage of this facet of these search techniques in our previous work [3]. Nevertheless, we do highlight some generals trends with regard to search effectiveness.

We start with recall, which measures the proportion of relevant results identified by each search technique. Fig. 5 graphs this value for each data set and search technique. As evidenced by the figure, recall varies considerably: no single search technique performs best on all three data sets. DISCOVER-II has the best overall performance followed by DPBF. Not surprisingly, most search techniques have little trouble identifying the relevant MONDIAL results due to MONDIAL's small size. Had BANKS managed to complete the entire query workload on MONDIAL, we have little doubt that its recall would be comparable to the other search techniques. The graph-based approaches' recall drops precipitously for Wikipedia, which is likely due to the amount of text appearing in Wikipedia articles.

12. STAR is the only search technique that substantially improves with additional memory for IMDb and Wikipedia. Other search techniques (e.g., BANKS-II, DPBF, and BLINKS) see modest improvement for the MONDIAL data set.

13. The size of the IMDb data graph is approximately 25 times the size of the relational data. Extrapolating to the complete IMDb, we would need hundreds of GBs of memory just to store the data graph!

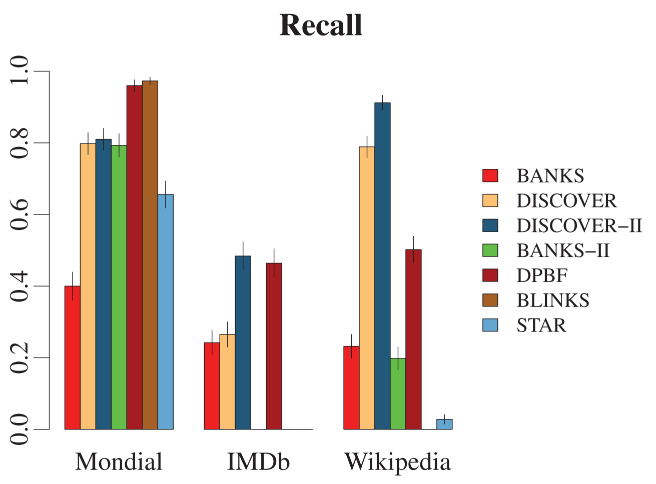


Fig. 5. Recall ($\in [0,1]$) measured across the various search techniques and data sets. The retrieval depth is 100 results. Higher bars are better, and the error bars denote the standard error of the mean.

DISCOVER-II's IR-style scoring function appears to help it rank relevant results within the top-100 results returned.

Fig. 6 graphs precision at k. P@1 is comparable to the number of relevant results ranked first by a search technique; this metric has been used in a number of previous evaluations [27], [28], [29]. The figure reveals two interesting trends. First, effectiveness drops as the retrieval depth increases although this result might be an artifact of having fewer than ten relevant results for some queries. It is also well-known in the IR community that improving recall typically diminishes precision. Second, the values are much lower than those reported in previous evaluations of these search techniques but more consistent with results from established IR evaluation venues (e.g., TREC).

In terms of overall search effectiveness (MAP in Fig. 7), the various search techniques vary widely. Not surprisingly, effectiveness is highest for our smallest data set. The best systems, DPBF and BLINKS, perform exceedingly well, which mirrors their excellent recall (see Fig. 5). We note that these scores are considerably higher than those that appear in IR venues (e.g., TREC), which likely reflects the small size of the MONDIAL database. Search effectiveness falls significantly for larger data sets. Unlike performance, which is generally consistent among systems, search effectiveness differs considerably. For example, DISCOVER-II performs poorly (relative to the other ranking schemes) for MONDIAL but proffers the greatest search effectiveness on Wikipedia. Hence, a ranking scheme that performs well for MONDIAL queries is not necessarily ideal for other data sets. It is important to balance performance concerns with a consideration of search effectiveness when designing relational keyword search systems.

5 Discussion

Overall, the performance of existing relational keyword search systems is disappointing, particularly with regard to the number of queries completed successfully in our query workload (see Table 5). We were especially surprised by the number of exceptions that we witnessed. Even if we ignore the exceptions related to memory consumption, the execution times of the search techniques are unacceptable for many queries. Our benchmark comprises real-world data sets and realistic queries so we believe that our results are more representative than many previous evaluations reported in the literature. Although additional search techniques that claim to address these runtime performance issues have been proposed, there is no indication that other proposed search techniques will outperform those in our evaluation.[14] We do acknowledge that deviations among implementations may account for the discrepancy between our results and those previously reported in the literature. For example, some prior evaluations enforce a much lower limit for execution time, which improves performance at the risk of lessening search effectiveness.

Because our larger execution times might only reflect our choice to use larger data sets, we focus on two concerns that we have related to memory utilization. Nevertheless, we start by mentioning the following two caveats: 1) the amount of memory given each search technique exceeds all prior evaluations except for BLINKS's evaluation, which used 4 GB of memory, and 2) our reimplementations of search techniques scaled better than any implementation that we were given from other researchers.

No search techniques admit to having a large memory requirement. In fact, memory consumption during a search has not been the focus of any previous evaluation. To the best of our knowledge, only two papers [10], [30] have been published in the literature that make allowances for a data graph that does not fit entirely within main memory. Given that most existing evaluations focus on performance, handling large data graphs (i.e., those that do not fit within main memory) should be well-studied. Relying on virtual memory and paging is no panacea to this problem because the operating system's virtual memory manager will induce much more I/O than algorithms designed for large graphs [30] as evidenced by the number of timeouts caused by severe thrashing in our experiments. Kasneci et al. [10] show that storing the graph on disk can also be extremely expensive for algorithms that touch a large number of nodes and edges.

Our results also seriously question the scalability of these search techniques. Although the amount of memory used in our experiments is small by today's standards so too are the data sets. More worrying is the fact that our IMDb data set is more than 25 times smaller than the entire IMDb. Without additional research into high-performance algorithms that maintain an acceptable memory footprint, relational keyword search techniques will be unable to search even moderately sized databases and will never be suitable for large databases like social networks or medical health records.

Precomputing search results proffers an alternative to the approaches compared in this paper, but precomputing search results is enormously expensive both in up-front costs and in the size of the resulting index [6], [7]. Other important factors such as keeping these materialized views

14. For example, BLINKS's order of magnitude improvement in execution time failed to materialize, and BLINKS consumes approximately two orders of magnitude more memory than BANKS-II.

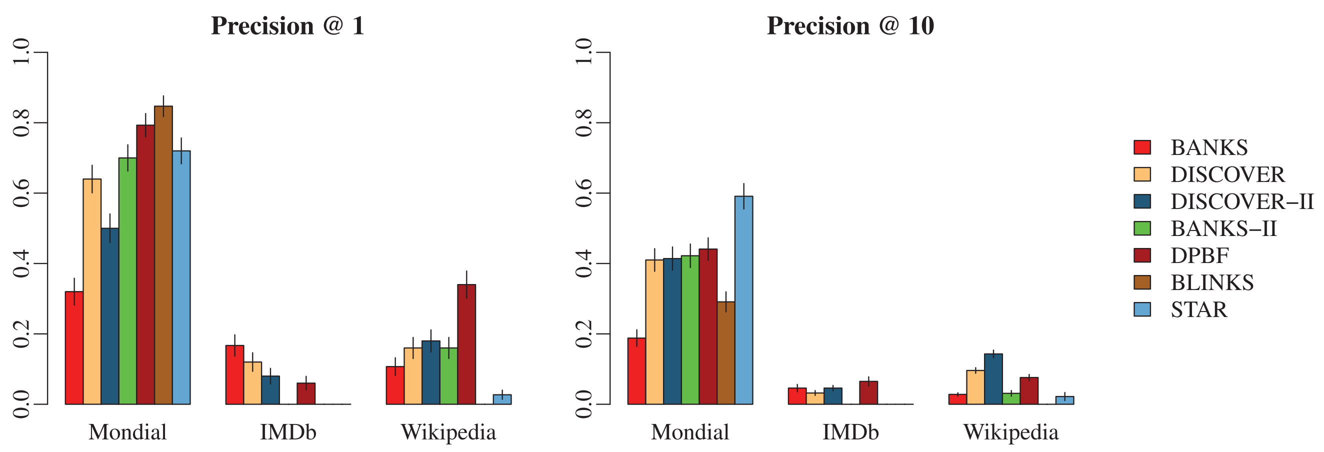


Fig. 6. Precision @ k ($\in [0, 1]$) measured across the various search techniques and data sets with a retrieval depth of 100. Higher bars are better, and the error bars denote the standard error of the mean.

consistent with their underlying data are rarely addressed, and some search techniques must rebuild their index *from scratch* in response to updates (i.e., there is no efficient scheme to update the index incrementally). While it would have been interesting to include these search techniques in our evaluation, the underlying approach is considerably different, which could make the results difficult to interpret. For example, an offline indexing technique might be ideal for read-only databases but might be completely inappropriate for write-intensive scenarios. More importantly, there is no evidence in the existing literature that the evaluation of these search techniques is more robust than those included in our experiments; we hypothesize that many more issues would be raised (e.g., the exponential growth in index size) had we included such approaches in our evaluation.

5.1 Threats to Validity

Our results naturally depend upon our evaluation benchmark. While we would like our experiments to include additional test collections created by other researchers, our benchmark is currently the only publicly available collection of data sets, queries, and relevance judgments.

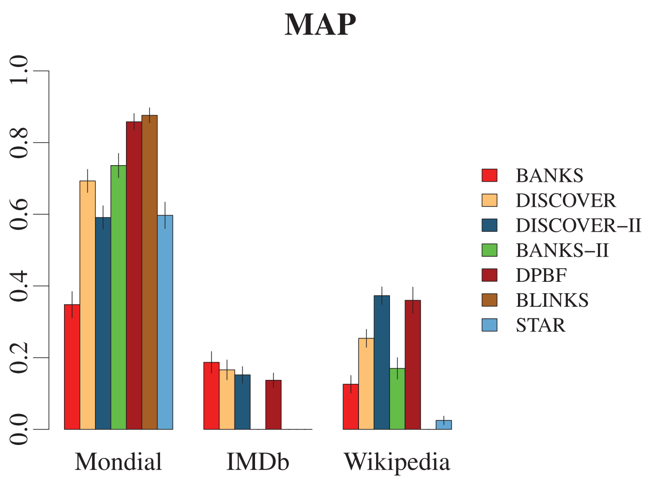


Fig. 7. MAP ($\in [0, 1]$) measured across the various search techniques and data sets. The retrieval depth is 100 results. Higher bars are better, and the error bars denote the standard error of the mean.

One implementation issue that does impact the results for the graph-based search techniques is the graph data structure. All the implementations in our evaluation use the JGraphT library,[15] which is designed to scale to millions of vertices and edges. Nevertheless, JGraphT relies upon dynamically sized collections, and an array-based graph implementation (e.g., Kacholia et al.'s [8]) can greatly reduce memory utilization at the cost of additional overhead to keep the data graph consistent with the underlying database. These issues have not been investigated in prior work.

6 Related Work

Existing evaluations of relational keyword search techniques are ad hoc with little standardization. Webber [5] summarizes existing evaluations with regards to search effectiveness. Our previous work [3] compares relational keyword search techniques with regard to search effectiveness but does not consider runtime performance. Baid et al. [6] assert that many existing keyword search techniques have unpredictable performance due to unacceptable response times or fail to produce results even after exhausting memory. Our results—particularly the large memory footprint of the systems—confirm this claim.

A number of relational keyword search systems have been published beyond those included in our evaluation. Chen et al. [4] and Chaudhuri and Das [31] both presented tutorials on keyword search in databases. Yu et al. [32] provide an excellent overview of relational keyword search techniques.

Liu et al. [27] and SPARK [28], [29] both propose schema-based approaches with modified scoring functions. SPARK [28], [29] introduces a skyline sweep algorithm to minimize the total number of database probes during a search. Golenberg et al. [24] provide an algorithm that enumerates results in approximate order by height with polynomial delay. Dalvi et al. [30] consider keyword search on graphs that cannot fit within main memory. Qin et al. [33] further pursue efficient query processing by exploring semi-joins. Baid et al. [6] suggest terminating the search after a

15. http://jgrapht.org/.

predetermined period of time and allowing the user to guide further exploration of the search space. Bicer et al. [34] and Coffman and Weaver [35] both investigate ranking schemes to improve search effectiveness. Mass and Sagiv [36] propose using language models to assign query-dependent weights to a graph.

Other researchers have suggested indexing search results offline to improve runtime performance. Su and Widom [7] reuse the full-text search capabilities of the underlying database. EASE [13] indexes all r-radius Steiner graphs that might form results for a keyword query. SAINT [37], [38] answers queries by combining tuples related by foreign key references. CSTree [15] uses compact Steiner trees to answer search queries more efficiently. One downside to these approaches is that the maximum distance between search terms is limited a priori: if the terms are not closely related, many offline search techniques will fail to return any results.

In general, the evaluations of proposed search techniques do not investigate important issues related to performance (e.g., handling data graphs that do not fit within main memory). Many evaluations are also contradictory, for the reported performance of each system varies greatly between different evaluations. For the offline approaches, the size of the index can exceed the size of the original database by an order of magnitude [6], [7]. Our experimental results question the validity of many previous evaluations, and we believe our benchmark is more robust and realistic with regards to the retrieval tasks than the workloads used in other evaluations.

7 Conclusion

Unlike many evaluations reported in the literature, ours investigates the overall, end-to-end performance of relational keyword search techniques. Hence, we favor a realistic query workload instead of a larger workload with queries that are unlikely to be representative (e.g., queries created by randomly selecting terms from the data set).

Our experimental results do not reflect well on existing relational keyword search techniques. Runtime performance is unacceptable for most search techniques. Memory consumption is also excessive for many search techniques. Our experimental results question the scalability and improvements claimed by previous evaluations. These conclusions are consistent with previous evaluations that demonstrate the poor runtime performance of existing search techniques as a prelude to a newly-proposed approach.

7.1 Future Work

Further research is unquestionably necessary to investigate the myriad of experimental design decisions that have a significant impact on the evaluation of relational keyword search systems. For example, our results indicate that existing systems would be unable to search the entire IMDb database, which underscores the need for a progression of data sets that will allow researchers to make progress toward this objective. Creating a subset of the original data set is common, but we are not aware of any work that identifies how to determine if a subset is representative of the original data set. In addition, different research groups often have different schemas for the same data (e.g., IMDb), but the effect of different database schemas on experimental results has also not been studied.

Our results should serve as a challenge to this community because little previous work has acknowledged these challenges. Moving forward, we must address several issues. First, we must design algorithms, data structures, and implementations that recognize that main memory is limited. Search techniques must manage their memory utilization efficiently, swapping data to and from disk as necessary. Such implementations are unlikely to have performance characteristics that are similar to existing approaches but must be used if relational keyword search systems are to scale to large data sets (e.g., hundreds of millions of tuples). Second, evaluations should reuse data sets and query workloads to provide greater consistency of results, for even our results vary widely depending on which data set is considered. Having the community coalesce behind reusable test collections would facilitate better comparison among systems and improve their overall evaluation [5]. Fortunately, our evaluation benchmark is beginning to gain traction in this area as evidenced by others' adoption of it for their evaluations [34], [36], [39]. Third, the practice of researchers reimplementing search techniques may account for some evaluation discrepancies. Making the original source code (or a binary distribution that accepts a database URL and query as input) available to other researchers would greatly reduce the likelihood that observed differences are implementation artifacts.

Acknowledgments

Michelle McDaniel contributed to preliminary research and provided feedback on drafts of this paper. The authors thank He et al. and Kasneci et al. for implementations of search techniques used in our experiments. This work was completed while J. Coffman was at the University of Virginia.

References

[1] D. Fallows, "Search Engine Use," technical report, Pew Internet and Am. Life Project, http://www.pewinternet.org/Reports/2008/Search-Engine-Use.aspx. Aug. 2008.

[2] comScore, "Global Search Market Grows 46 Percent in 2009," http://www.comscore.com/Press_Events/Press_Releases/2010/1/Global_Searc h_Market_Grows_46_%_in_2009, Jan. 2010.

[3] J. Coffman and A.C. Weaver, "A Framework for Evaluating Database Keyword Search Strategies," *Proc. 19th ACM Int'l Conf. Information and Knowledge Management (CIKM '10)*, pp. 729-738, Oct. 2010.

[4] Y. Chen, W. Wang, Z. Liu, and X. Lin, "Keyword Search on Structured and Semi-Structured Data," *Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '09)*, pp. 1005-1010, June 2009.

[5] W. Webber, "Evaluating the Effectiveness of Keyword Search," *IEEE Data Eng. Bull.*, vol. 33, no. 1, pp. 54-59, Mar. 2010.

[6] A. Baid, I. Rae, J. Li, A. Doan, and J. Naughton, "Toward Scalable Keyword Search over Relational Data," *Proc. VLDB Endowment*, vol. 3, no. 1, pp. 140-149, 2010.

[7] Q. Su and J. Widom, "Indexing Relational Database Content Offline for Efficient Keyword-Based Search," *Proc. Ninth Int'l Database Eng. and Application Symp. (IDEAS '05)*, pp. 297-306, July 2005.

[8] V. Kacholia, S. Pandit, S. Chakrabarti, S. Sudarshan, R. Desai, and H. Karambelkar, "Bidirectional Expansion For Keyword Search on Graph Databases," *Proc. 31st Int'l Conf. Very Large Data Bases (VLDB '05)*, pp. 505-516, Aug. 2005.

[9] H. He, H. Wang, J. Yang, and P.S. Yu, "BLINKS: Ranked Keyword Searches on Graphs," *Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '07),* pp. 305-316, June 2007.

[10] G. Kasneci, M. Ramanath, M. Sozio, F.M. Suchanek, and G. Weikum, "STAR: Steiner-Tree Approximation in Relationship Graphs," *Proc. Int'l Conf. Data Eng. (ICDE '09),* pp. 868-879, Mar. 2009.

[11] G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S. Sudarshan, "Keyword Searching and Browsing in Databases Using BANKS," *Proc. 18th Int'l Conf. Data Eng. (ICDE '02),* pp. 431-440, Feb. 2002.

[12] B. Ding, J.X. Yu, S. Wang, L. Qin, X. Zhang, and X. Lin, "Finding Top-k Min-Cost Connected Trees in Databases," *Proc. 23rd Int'l Conf. Data Eng. (ICDE '07),* pp. 836-845, Apr. 2007.

[13] G. Li, B.C. Ooi, J. Feng, J. Wang, and L. Zhou, "EASE: An Effective 3-in-1 Keyword Search Method for Unstructured, Semi-Structured and Structured Data," *Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '08),* pp. 903-914, June 2008.

[14] L. Qin, J. Yu, L. Chang, and Y. Tao, "Querying Communities in Relational Databases," *Proc. IEEE Int'l Conf. Data Eng. (ICDE '09),* pp. 724-735, Mar. 2009.

[15] G. Li, J. Feng, X. Zhou, and J. Wang, "Providing Built-in Keyword Search Capabilities in RDBMS," *The VLDB J.,* vol. 20, pp. 1-19, Feb. 2011.

[16] V. Hristidis and Y. Papakonstantinou, "DISCOVER: Keyword Search in Relational Databases," *Proc. 28th Int'l Conf. Very Large Data Base (VLDB '02),* pp. 670-681, Aug. 2002.

[17] V. Hristidis, L. Gravano, and Y. Papakonstantinou, "Efficient IR-Style Keyword Search over Relational Databases," *Proc. 29th Int'l Conf. Very Large Data Bases (VLDB '03),* pp. 850-861, Sept. 2003.

[18] A. Singhal, J. Choi, D. Hindle, D. Lewis, and F. Pereira, "AT&T at TREC-7," *Proc. Seventh Text REtrieval Conf. (TREC-7),* pp. 239-252, Nov. 1999.

[19] S.E. Dreyfus and R.A. Wagner, "The Steiner Problem in Graphs," *Networks,* vol. 1, no. 3, pp. 195-207, 1971.

[20] G. Reich and P. Widmayer, "Beyond Steiner's Problem: A VLSI Oriented Generalization," *Proc. 15th Int'l Workshop Graph-Theoretic Concepts in Computer Science,* pp. 196-210, 1990.

[21] W. May, "Information Extraction and Integration with Florid: The Mondial Case Study," Technical Report 131, Universität Freiburg, Institut für Informatik, 1999.

[22] G. Pass, A. Chowdhury, and C. Torgeson, "A Picture of Search," *Proc. First Int'l Conf. Scalable Information Systems (InfoScale '06),* May 2006.

[23] J. Coffman and A.C. Weaver, "What Are We Searching For? Analyzing User Objectives When Searching Relational Data," *Proc. Workshop Web Search Click Data (WSCD '12),* Feb. 2012.

[24] K. Golenberg, B. Kimelfeld, and Y. Sagiv, "Keyword Proximity Search in Complex Data Graphs," *Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '08),* pp. 927-940, June 2008.

[25] J. Coffman and A.C. Weaver, "An Empirical Performance Evaluation of Relational Keyword Search Systems," Technical Report CS-2011-07, Univ. of Virginia, 2011.

[26] X. Yang, C.M. Procopiuc, and D. Srivastava, "Summarizing Relational Databases," *Proc. VLDB Endowment,* vol. 2, pp. 634-645, Aug. 2009.

[27] F. Liu, C. Yu, W. Meng, and A. Chowdhury, "Effective Keyword Search in Relational Databases," *Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '06),* pp. 563-574, June 2006.

[28] Y. Luo, X. Lin, W. Wang, and X. Zhou, "SPARK: Top-k Keyword Query in Relational Databases," *Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '07),* pp. 115-126, June 2007.

[29] Y. Luo, W. Wang, X. Lin, X. Zhou, J. Wang, and K. Li, "SPARK2: Top-k Keyword Query in Relational Databases," *IEEE Trans. Knowledge and Data Eng.,* vol. 23, no. 12, pp. 1763-1780, Dec. 2011.

[30] B.B. Dalvi, M. Kshirsagar, and S. Sudarshan, "Keyword Search on External Memory Data Graphs," *Proc. VLDB Endowment,* vol. 1, no. 1, pp. 1189-1204, 2008.

[31] S. Chaudhuri and G. Das, "Keyword Querying and Ranking in Databases," *Proc. VLDB Endowment,* vol. 2, pp. 1658-1659, Aug. 2009.

[32] J.X. Yu, L. Qin, and L. Chang, *Keyword Search in Databases,* first ed. Morgan and Claypool Publishers, 2010.

[33] L. Qin, J.X. Yu, and L. Chang, "Keyword Search in Databases: The Power of RDBMS," *Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '09),* pp. 681-694, June 2009.

[34] V. Bicer, T. Tran, and R. Nedkov, "Ranking Support for Keyword Search on Structured Data Using Relevance Models," *Proc. 20th ACM Int'l Conf. Information and Knowledge Management (CIKM '11),* pp. 1669-1678, 2011.

[35] J. Coffman and A.C. Weaver, "Learning to Rank Results in Relational Keyword Search," *Proc. 20th ACM Int'l Conf. Information and Knowledge Management (CIKM '11),* Oct. 2011.

[36] Y. Mass and Y. Sagiv, "Language Models for Keyword Search over Data Graphs," *Proc. Fifth ACM Int'l Conf. Web Search and Data Mining (WSDM '12),* pp. 363-372, Feb. 2012.

[37] G. Li, J. Feng, and L. Zhou, "RETUNE: Retrieving and Materializing Tuple Units for Effective Keyword Search over Relational Databases," *Proc. Int'l Conf. Conceptual Modeling,* pp. 469-483, 2008.

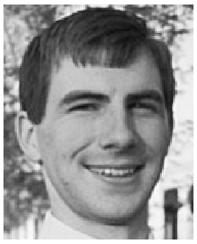
[38] J. Feng, G. Li, and J. Wang, "Finding Top-k Answers in Keyword Search over Relational Databases Using Tuple Units," *Trans. Knowledge and Data Eng.,* vol. 23, no. 12, pp. 1781-1794, Dec. 2011.

[39] S. Yogev, H. Roitman, D. Carmel, and N. Zwerdling, "Towards Expressive Exploratory Search over Entity-Relationship Data," *Proc. 21st Int'l Conf. Companion on World Wide Web (WWW '12 Companion),* pp. 83-92, 2012.

[40] N. Fuhr, N. Gövert, G. Kazai, and M. Lalmas, "INEX: Initiative for the Evaluation of XML Retrieval," *Proc. SIGIR Workshop XML and Information Retrieval,* Aug. 2002.

[41] E.M. Voorhees, "The Philosophy of Information Retrieval Evaluation," *Proc. Second Workshop Cross-Language Evaluation Forum on Evaluation of Cross-Language Information Retrieval Systems (CLEF '01),* pp. 355-370, 2002.

[42] C. Cleverdon, "The Cranfield Tests on Index Language Devices," *Readings in Information Retrieval,* K.S. Jones and P. Willett, eds., pp. 47-59, Morgan Kaufmann, 1997.



Joel Coffman received the BS degree in computer science from Furman University and the MS and PhD degrees in computer science from the University of Virginia. His research interests include databases and information retrieval and a desire to transition research technology into deployed tools. He is a member of the IEEE.

Alfred C. (Alf) Weaver received the PhD degree in computer science from the University of Illinois in 1976. He is currently a professor of computer science and director of the University of Virginia's Applied Research Institute. His research areas include telemedicine, biotelemetrics, database search, and crowdsourcing. He is a fellow of the IEEE.