DualIso: Scalable Subgraph Pattern Matching On Large Labeled Graphs SUPPLEMENT

Matthew Saltz, Ayushi Jain, Abhishek Kothari, Arash Fard, John A. Miller and Lakshmish Ramaswamy

Computer Science Department

University of Georgia,
Athens, GA, USA, 30602

This supplement to the paper provides all figures and substantial code listings which would not fit into the paper due to page limitations.

Permalink: http://cobweb.cs.uga.edu/~jam/scalation_1.1/README.html#papers

Roadmap: The supplement is organized in the same manner as the main paper i.e. the section names in the main paper corresponds to the same section in the supplement for the easy reference.

Note: In many cases, text from the main paper is included and or extended in order to provide context for the figures, tables, and code listings contained in this supplement. The figure numbers may get changed as some more figures are added in this supplement for detailed explanation.

II. BACKGROUND & TERMINOLOGY

A. Graph Pattern Matching

The goal of graph pattern matching is to find all subgraphs of a large graph, called the data graph, that are similar to a query graph structurally and semantically. To consider why this might be useful, look again at the graph displayed in Figure 1. Suppose a user of Facebook has just moved to Lyon, France from the United States and is looking for someone to accompany him to the U2 concert there. Thus, he wants to find out if any of his friends are friends with a person who lives in Lyon and who likes U2. Given that Facebook can be modeled as a graph like Figure 1, this question can be modeled as a graph query, as shown in Figure 2. In this query, the capital letters represent variables that the user would like filled in. In this case, using the graph in Figure 1, the query would return X = John and Y = Jim. In fact, Facebook Graph Search2 attempts to allow users to do just that, though at the moment it only has limited querying capabilities.

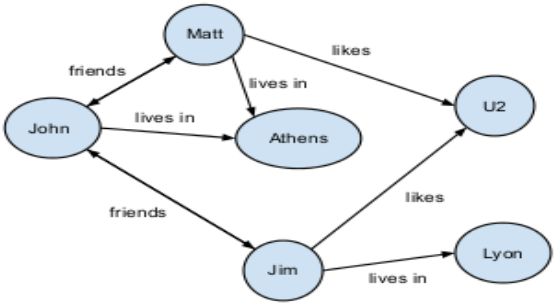


Fig. 1: A possible social graph.

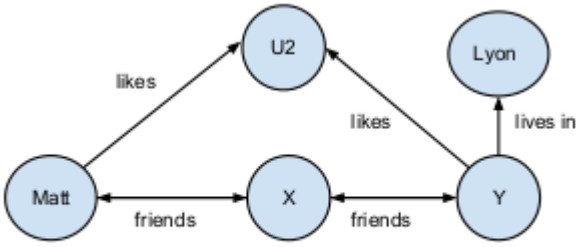


Fig. 2: A possible Facebook graph query.

B. Subgraph Isomorphism

A common way to define the criteria for exact matching is to use the definition of subgraph isomorphism. A graph is subgraph isomorphic to another if the first graph matches a subgraph of the second structurally and semantically. A query graph and its subgraph isomorphic matches from a data graph are shown in Figure 1(of the main paper) and Table I (of the main paper).

The objective of the subgraph isomorphism problem is to find all subgraph isomorphic matches of Q in G. This problem is known to be NP-hard [4], but with a fixed query size can be computed in polynomial time with respect to the query size. Looking at Table I, it is easy to see why the subgraph isomorphism problem is intractable in the worst case. Suppose we have an unlabeled graph with n vertices, and every vertex is connected to every other vertex. Now suppose we use that same graph as a query graph. Then every possible permutation of the vertices in the data graph is subgraph isomorphic to the query graph. This means that there are n! matches, and it would take at least O(n!) time just to enumerate all of them. Luckily, if the query is size n_q, then the number of matches is at most $n!/ (n-n_q)!$, which is $O(n^{\wedge}n_q)$. In practice, because the query graph is often much smaller than the data graph and because the vertex labels reduce the number of possible matches, finding all subgraph isomorphisms matches can be feasible.

III. TECHNIQUES FOR GRAPH PATTERN MATCHING

VF2 Algorithm, Step 1 of the VF2 algorithm is the same as Step 1 of Ullmann's algorithm. However, VF2 utilizes a "build up" rather than a "prune down" approach, and so Step 2 does not exactly apply to VF2. Rather, search paths are eliminated as an ongoing process during search. From the initial set of feasible matches, VF2 begins by adding a matching (query vertex, data vertex) pair to a partial match set, before performing a depth first search through the data graph and the query graph simultaneously starting at these vertices. At each stage it eliminates search paths based on local information. Further details of the algorithm can be found in [16].

A. Related Work

The field of subgraph pattern matching has been studied for decades [1]. As previously mentioned, pattern matching differs between two different kinds of graph databases. One consists of a collection of small-medium size graphs (the graph-transactional case), whereas the other consists of a single large graph [3]. In the graph transactional setting, the focus of a pattern matching algorithm is to find all graphs in the database that contain a subgraph similar to a query graph (and possibly to return those subgraphs) [3]. In a single graph setting, which we discuss, the goal is to find all subgraphs of a data graph that are similar to a query graph. Furthermore, the field is divided into exact and inexact matching, which we discussed earlier.

IV. PRESENTATION OF ALGORITHM

B. Pruning Algorithms

1) Simple Simulation:

The pruning procedure used for our subgraph isomorphism algorithm is based on a concept known as graph simulation [12], [16]. The basic version of graph simulation, simple simulation, is conceptually similar to the refinement procedure used by Ullmann's algorithm for pruning. The simple simulation condition alone (Equation 1) fails to prune large graphs well enough to be used effectively in obtaining all subgraph isomorphic matches. An example of simple simulation on a graph can be seen in Figure 2 (of the main paper). Note that vertex 0 is in Φ(1) despite the fact that it does not have a vertex in Φ(0) as a parent. Using simple simulation pruning technique, we have implemented a subgraph isomorphism algorithm, GraphSimIso, and the performance evaluation is shown in the experiment section V.

The conditions in Equation 1 can be adapted to create an algorithm as seen in the pseudocode [26]. Because $|\Phi(0)| + ... + |\Phi(|Vq| - 1)| \leq |V||Vq|$, the outer while loop in line 3 may execute at most $|V||Vq|$ times. This would correspond to a scenario in which only one vertex is removed with each iteration of the loop. The next two for loops (lines 5-6) execute a total of $|Eq|$ times, and there are at most $|V|$ vertices in $|\Phi(u)|$ for any $u \in Vq$, which creates a bound on the innermost for loop (line 7). The intersection operation in line 8 takes at most $|V|$ steps, and so the total time complexity is $O(|Eq||Vq||V|^3)$. Though [26] presents an algorithm to reduce the time complexity to be quadratic in $|V|$, it requires a parent list along with an adjacency list to describe all incoming edges into each vertex, which nearly doubles the memory requirement and may be infeasible for large graphs.

```
1:  procedure SIMPLESIM(G, Q, Φ):
2:      changed ← true
3:      while changed do
4:          changed ← false
5:          for u ← V_q do
6:              for u' ← Q.adj(u) do
7:                  for v ← Φ(u) do
8:                      if G.adj(v) ∩ Φ(u') = ∅ then
9:                          remove v from Φ(u)
10:                         if Φ(u) = ∅ then
11:                             return empty Φ
12:                         changed ← true
13:     return Φ
14: end procedure
```

Pseudocode for the graph simulation algorithm

C. Dual-based Isomorphism

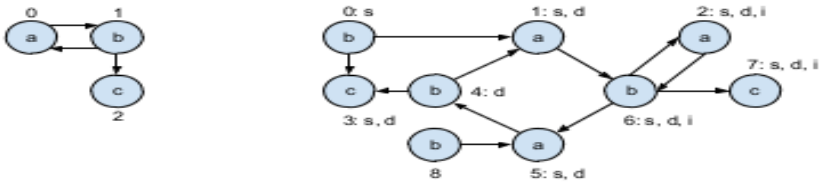


Figure 2: An example of a query graph (left) and its matches via simple, dual, and isomorphism in a data graph (right). Simple matches are denoted by an 's', dual matches by a 'd', and isomorphic matches by an 'i'. Note that any vertex that matches via isomorphism also matches via dual, which in turn matches simple.

To picture the operation of the algorithm, consider again the example in Figure 2. The first step retrieves all feasible matches based on labels, which returns

$$\Phi(0) = \{1, 2, 5\},$$
$$\Phi(1) = \{0, 4, 6, 8\}, \text{ and}$$
$$\Phi(2) = \{3, 7\}.$$

Following this, dual simulation is performed on Φ, yielding the vertices marked with 'd' in Figure 2. This eliminates nodes 0 and 8 from $\Phi(1)$. Next, the search process begins by creating a version Φ' of Φ such that $\Phi'(0) = 1$. Thus, when dual simulation is performed again on Φ', vertex 4 loses its necessary parent (vertex 5) and vertex 6 loses its necessary child (vertex 2). Consequently, these vertices are removed from $\Phi'(1)$, which becomes empty, and so dual simulation returns no result, causing the algorithm to backtrack. When the same process is undergone with $\Phi'(0) = \{2\}$, however, dual simulation only removes vertex 4 from $\Phi'(1)$ and vertex 3 from $\Phi'(2)$, leaving $\Phi'(0) = \{2\}$, $\Phi'(1) = \{6\}$, and $\Phi'(2) = \{7\}$, which contains only the vertices found in the isomorphic mapping. It recurses with depth = 1, finds that all vertices still satisfy dual simulation, recurses again until depth = 3 and then returns a match. It backtracks up to the top level, and when vertex 5 is isolated in $\Phi'(0)$, dual simulation returns no result. The algorithm completes having found the only isomorphic match.

1) Proof of Correctness: The correctness of the algorithm is now demonstrated.

Lemma 4.1: The initial Φ contains all possible subgraph isomorphic matches for each vertex in the query graph, and furthermore, no subgraph isomorphic matches are eliminated in the process of search.

Proof: The initial retrieval of feasible matches is based purely on labels, and all vertices are included that satisfy the label condition of subgraph isomorphism for each query vertex. Thus, no valid vertices are precluded in this way. If a subgraph of the data graph is isomorphic to the query graph, then condition 2 of Definition 2.2 holds for the vertices and edges in that subgraph. This implies that the dual simulation conditions are also satisfied for these vertices, and therefore, dual simulation never eliminates a vertex that occurs in a subgraph isomorphic match.

Lemma 4.2: All results of the dual-based isomorphism algorithm are subgraph isomorphic to the query graph. Proof: In the first case, when the dual-based isomorphism algorithm returns no matches, by Lemma 4.1, there are no subgraph isomorphic matches in the data graph. In the other case, when maximum depth is reached, if Φ is non-empty, the following conditions hold:

1) Each Φ(i) contains a single unique vertex. Thus, Φ is a bijection from the query graph onto a subset of the vertices of the data graph. Call this subset V'.
2) For each vertex u in the query graph, lq (u) = l(Φ(u)) (due to the condition on the initial retrieval of feasible matches).
3) The condition in Equation 1 holds. Because Φ(u) and Φ(u') are now unique vertices, rather than sets, this condenses to:

$$\forall (u, u') \in E_q , (\Phi(u), \Phi(u')) \in E.$$

Call the set of all such edges in the data graph E'. Then we have that $(u, u) \in E_q$ if and only if $(\Phi(u), \Phi(u')) \in E'$.
Letting l' be the projection of l onto V', by Definition 2.2, the subgraph G'(V', E', l') is necessarily an isomorphic match to Q with bijection $\Phi : V_q \rightarrow V'$.

Lemma 4.3: The algorithm always successfully terminates.

Proof: See runtime analysis in Section IV-C2.

Theorem 4.1: The algorithm for dual-based isomorphism correctly finds all subgraph isomorphic matches of a query graph in a data graph.

Proof: This follows directly from the previous lemmas.

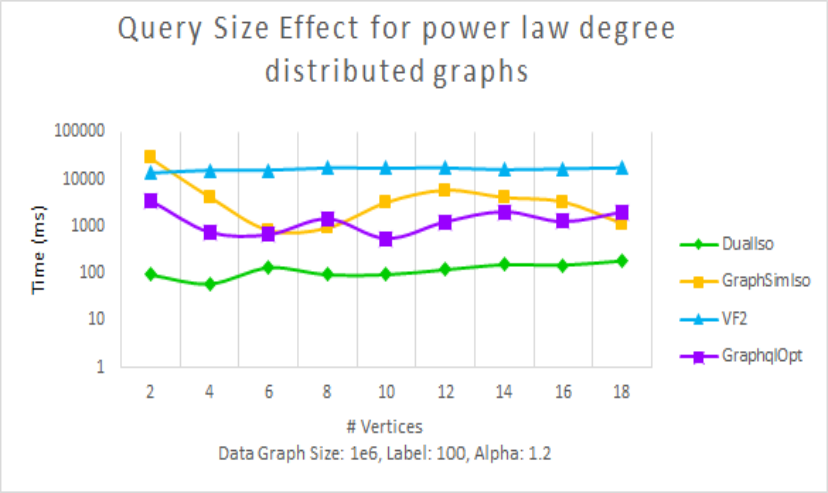
2) Runtime Analysis: Following the logic in Section II-A2, the worst-case number of matches given a query graph Q(Vq , Eq , lq) a data graph G(V, E, l) is O(|V |^|Vq |). Dual simulation must then be executed on each of these O(|Vq |) times (the maximum depth of the search tree) in the worst case, yielding a time complexity of O(|V |^(|Vq |+3) |Vq |^2 |Eq |).

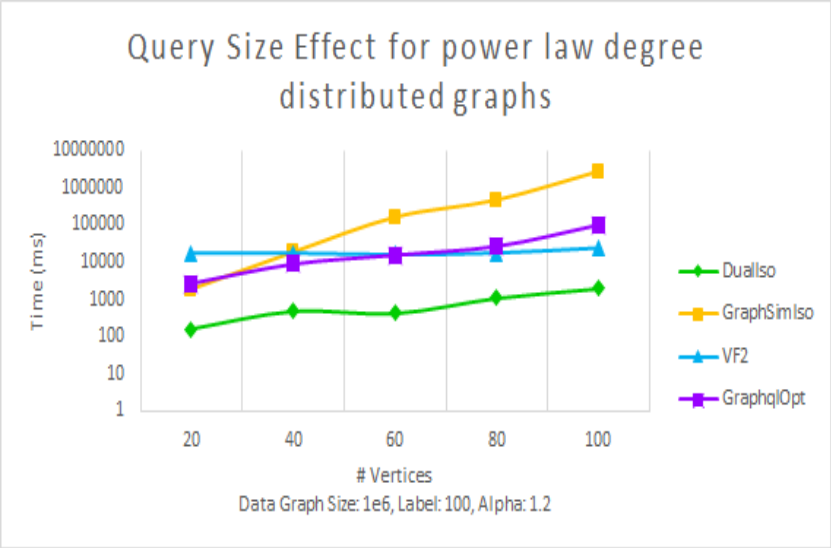
V. EXPERIMENTATION

C. Synthesized Data

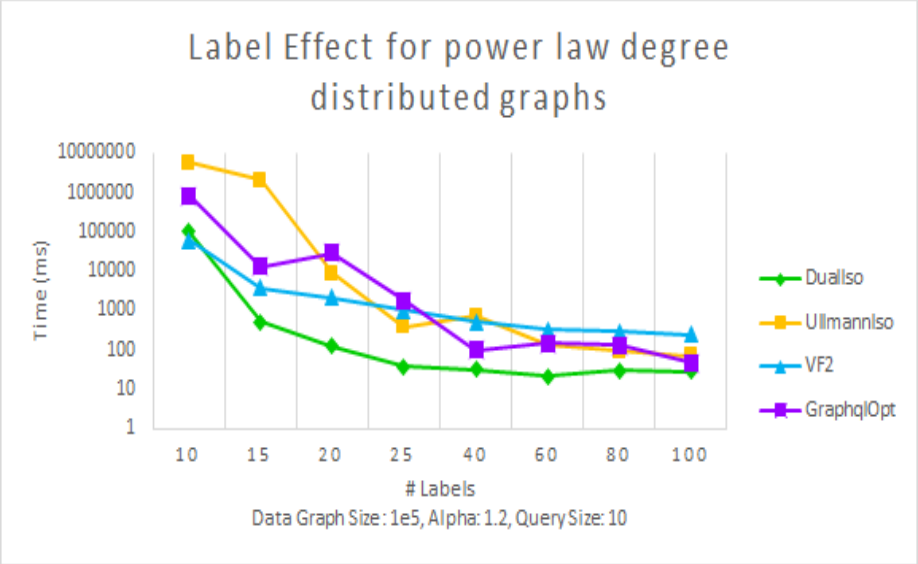
This portion of the experimentation was dedicated to determining the effects of various factors on the runtime of DualIso, GraphSimIso, GraphqlOpt and VF2.

2) Effect of Query Size: In this section of the supplement, we have shown the results for power law data graph, which shows the same behavior as uniform degree distribution data graph.





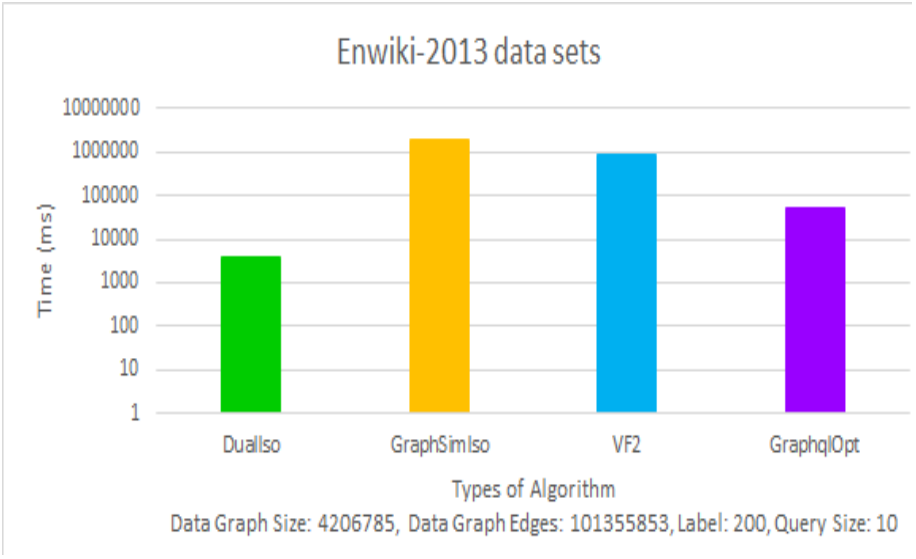
3) Effect of Labels: In this section of the supplement, we have shown the results for power law data graph, which shows the same behavior as uniform degree distribution data graph.



The effect of labels for power law degree distributed graph on runtime.

D. Real Data

2) Enwiki-2013 Dataset: The Enwiki-2013 dataset is a graph consisting of 4,206,785 vertices, 101,355,853 edges and we randomly assign 200 labels. This graph represents a snapshot of the English part of Wikipedia as of late February 2013. The titles of the pages are the identifiers and the edges are the links. The BFS queries were used to generate 20 queries each of size 10 and the number of edges based on $\alpha = 1.2$ as shown in the below figure.


Enwiki-2013 data sets
10000000
1000000
100000
10000
1000
100
10
1
Time (ms)
DualIso
GraphSimIso
VF2
GraphqlOpt
Types of Algorithm
Data Graph Size: 4206785, Data Graph Edges: 101355853, Label: 200, Query Size: 10

REFERENCES

[1] D. Conte, P. Foggia, C. Sansone, and M. Vento, "Thirty years of graph matching in pattern recognition," International journal of pattern recognition and artificial intelligence, vol. 18, no. 03, pp. 265–298, 2004.

[2] C. C. Aggarwal and H. Wang, Managing and mining graph data. Springer, 2010, vol. 40.

[3] B. Gallagher, "Matching structure and semantics: A survey on graph based pattern matching," AAAI FS, vol. 6, pp. 45–53, 2006.

[4] J. Lee, W.-S. Han, R. Kasperovics, and J.-H. Lee, "An in-depth comparison of subgraph isomorphism algorithms in graph databases," in Proceedings of the 39th international conference on Very Large Data Bases. VLDB Endowment, 2012, pp. 133–144.

[5] L. Dehaspe, H. Toivonen, and R. D. King, "Finding frequent substructures in chemical compounds." in KDD, vol. 98, 1998, p. 1998.

[6] M. Deshpande, M. Kuramochi, N. Wale, and G. Karypis, "Frequent substructure-based approaches for classifying chemical compounds," Knowledge and Data Engineering, IEEE Transactions on, vol. 17, no. 8, pp. 1036–1050, 2005.

[7] S. Kramer, L. De Raedt, and C. Helma, "Molecular feature mining in hiv data," in Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2001, pp.136–143.

[8] J. M. Kleinberg, R. Kumar, P. Raghavan, S. Rajagopalan, and A. S.Tomkins, "The web as a graph: Measurements, models, and methods," in Computing and combinatorics. Springer, 1999, pp. 1–17.

[9] S. Wasserman, Social network analysis: Methods and applications. Cambridge university press, 1994, vol. 8.

[10] W. Chen, Y. Wang, and S. Yang, "Efficient influence maximization in social networks," in Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2009, pp.199–208.

[11] H. He and A. K. Singh, "Graphs-at-a-time: query language and access methods for graph databases," in Proceedings of the 2008 ACM SIGMOD international conference on Management of data. ACM, 2008, pp. 405–418.

[12] H. Shang, Y. Zhang, X. Lin, and J. X. Yu, "Taming verification hardness: an efficient algorithm for testing subgraph isomorphism," Proceedings of the VLDB Endowment, vol. 1, no. 1, pp. 364–375, 2008.

[13] P. Zhao and J. Han, "On graph query optimization in large networks," Proceedings of the VLDB Endowment, vol. 3, no. 1-2, pp. 340–351, 2010.

[14] Z. Sun, H. Wang, H. Wang, B. Shao, and J. Li, "Efficient subgraph matching on billion node graphs," Proceedings of the VLDB Endowment, vol. 5, no. 9, pp. 788–799, 2012.

[15] S. Zhang, S. Li, and J. Yang, "Gaddi: distance index based subgraph matching in biological networks," in Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology. ACM, 2009, pp. 192–203.

[16] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento, "A (sub) graph isomorphism algorithm for matching large graphs," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 26, no. 10, pp. 1367–1372, 2004.

[17] J. R. Ullmann, "An algorithm for subgraph isomorphism," Journal of the ACM (JACM), vol. 23, no. 1, pp. 31–42, 1976.

[18] W. J. Christmas, J. Kittler, and M. Petrou, "Structural matching in computer vision using probabilistic relaxation," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 17, no. 8, pp. 749–764, 1995.

[19] S. Umeyama, "An eigendecomposition approach to weighted graph matching problems," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 10, no. 5, pp. 695–703, 1988.

[20] S. Ma, Y. Cao, W. Fan, J. Huai, and T. Wo, "Capturing topology in graph pattern matching," Proceedings of the VLDB Endowment, vol. 5, no. 4, pp. 310–321, 2011.

[21] M. U. Nisar, A. Fard, and J. A. Miller, "Techniques for graph analytics on big data," in Proceedings of the IEEE Big Data Congress, 2013.

[22] D. J. Cook and L. B. Holder, "Substructure discovery using minimum description length and background knowledge," arXiv preprint cs/9402102, 1994.

[23] H. Bunke and G. Allermann, "Inexact graph matching for structural pattern recognition," Pattern Recognition Letters, vol. 1, no. 4, pp. 245–253, 1983.

[24] A. Sanfeliu and K.-S. Fu, "A distance measure between attributed relational graphs for pattern recognition," Systems, Man and Cybernetics, IEEE Transactions on, no. 3, pp. 353–362, 1983.

[25] W.-S. Han, J. Lee, and J.-H. Lee, "Turbo iso: towards ultrafast and robust subgraph isomorphism search in large graph databases," in Proceedings of the 2013 international conference on Management of data. ACM,
2013, pp. 337–348. [26] M. R. Henzinger, T. A. Henzinger, and P. W. Kopke, "Computing simulations on finite and infinite graphs," in Foundations of Computer Science, 1995. Proceedings, 36th Annual Symposium on. IEEE, 1995, pp. 453–462.

[27] G. Csardi and T. Nepusz, "The igraph software package for complex network research," InterJournal, Complex Systems, vol. 1695, no. 5, 2006.

[28] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On power-law relationships of the internet topology," in ACM SIGCOMM Computer Communication Review, vol. 29, no. 4. ACM, 1999, pp. 251–262.

[29] R. Govindan and H. Tangmunarunkit, "Heuristics for internet map discovery," in INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, vol. 3. IEEE, 2000, pp. 1371–1380.

[30] A.-L. Barab´asi and R. Albert, "Emergence of scaling in random networks," science, vol. 286, no. 5439, pp. 509–512, 1999.

[31] R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tomkins, and E. Upfal, "Stochastic models for the web graph," in Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on. IEEE, 2000, pp. 57–65.

[32] R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins, "Extracting large-scale knowledge bases from the web," in VLDB, vol. 99. Citeseer, 1999, pp. 639–650.

[33] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener, "Graph structure in the web," Computer networks, vol. 33, no. 1, pp. 309–320, 2000.

[34] P. Boldi, B. Codenotti, M. Santini, and S. Vigna, "Structural properties of the african web," in The Eleventh International WWW Conference, vol. 66, 2002.

[35] S. Redner, "How popular is your paper? an empirical study of the citation distribution," The European Physical Journal B-Condensed Matter and Complex Systems, vol. 4, no. 2, pp. 131–134, 1998.

[36] L. A. Adamic and B. A. Huberman, "The web's hidden order," Communications of the ACM, vol. 44, no. 9, pp. 55–60, 2001.