Improving Algorithm Search Using the Algorithm Co-Citation Network

Suppawong Tuarob[†], Prasenjit Mitra[†‡] and C. Lee Giles[†‡]
[†] Computer Science and Engineering, [‡] Information Sciences and Technology
The Pennsylvania State University
University Park, PA 16802
suppawong@psu.edu, {pmitra, giles}@ist.psu.edu

ABSTRACT

Algorithms are an essential part of computational science. An algorithm search engine, which extracts pseudo-codes and their metadata from documents, and makes it searchable, has recently been developed as part of the Citeseer^X suite [3, 4]. However, this algorithm search engine only retrieves and ranks relevant algorithms solely on textual similarity. Here, we propose a method for using the algorithm co-citation network to infer the similarity between algorithms. We apply a graph clustering algorithm on the network for algorithm recommendation and make suggestions on how to improve the current Citeseer^X algorithm search engine.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]

Keywords

Algorithms, Clustering, Algorithm Co-Citation Network

1. INTRODUCTION

Computer science is often about algorithms. Searching for the right algorithms for a specific problem can be a challenging task, as there has not been a way for automatically interpreting the semantics of algorithms. Methods for searching for algorithms have been implemented by applying traditional search engine techniques on algorithm metadata. Bhatia et al.[3] developed an algorithm extractor which extracts pseudo-codes along with their metadata such as captions, reference sentences and synopses [1, 2]. Indexing such algorithm metadata makes it searchable. However, the search is done by text based matching of user queries with the metadata.

The ability to infer the similarity of algorithms could also be beneficial when searching for algorithms. One may want to know if there are other available algorithms which address the same problem as a known algorithm. For example, one might want to know if there are other algorithms that find shortest paths in a graph such as Dijkstra's algorithm. Researchers often would want to search for existing algorithms so that they can develop a better algorithm or use these algorithms as baselines for their experiments. Detection of the similarity of algorithms could also lead to the discovery of newly emerging algorithms which are not yet well known.

We hypothesize that the similarity between algorithms can be captured using an algorithm co-citation network. We generate the algorithm co-citation network from scientific documents in the Citeseer^X[1] repository. We apply a clustering algorithm on a sample subset of the algorithm co-citation network which gives groups of relevant algorithm-proposing documents. We evaluate the clustering results by varying the clustering granularity levels and measure the meaningfulness of each cluster.

2. OBSERVATIONS AND MOTIVATIONS

In an algorithm-proposing document, there is usually a paragraph (in the introduction or the related work sections) devoting to addressing past work. In this case, we are interested in algorithms used in previous documents. From reading multiple documents throughout our own research, we find that if multiple algorithms are mentioned in a paper, then it is likely that these algorithms address similar problems. From these observations, we hypothesize that, if two algorithms are co-cited multiple times, then there could exist a relationship between them which can be used to infer their similarity. Such a relationship could be represented using the algorithm co-citation graph, which is explained in a later section.

3. RELATED WORK

Even though the document co-citation has been extensively studied, to the best of our knowledge, there has not been any work on algorithm co-citation analysis. Hence, the related work that we present here are mostly related to the current Citeseer^X algorithm search engine.

The current Citeseer^X algorithm search engine uses an automated algorithm extractor to extract pseudo-code which appears as document-elements with captions in scientific documents. The automated algorithm extractor extracts related information of each pseudo-code such as its caption, reference sentences, and year of publication (if available). A pseudo-code reference sentence is a sentence in the document which mentions the pseudo-code. A synopsis is also generated as part of the metadata to provide an overview

[1] http://citeseerx.ist.psu.edu

JCDL'12, June 10–14, 2012, Washington, DC, USA.

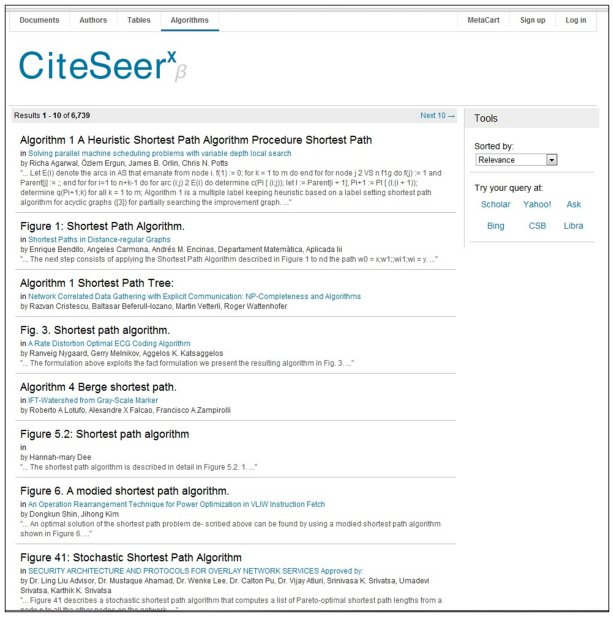

Figure 1: Citeseer^X algorithm search engine.

for each pseudo-code extracted[1]. A synopsis of a pseudo-code is generated from the set of its reference sentences, constructed by heuristics and machine learning techniques. All the extracted metadata information is then fed to the indexer. When a user inputs a query, TF-IDF based cosine similarity scores are computed to retrieve and rank the relevant pseudo-codes. Each search result gives a pointer to the document where the pseudo-code resides. Figure 1 shows the results returned by the Citeseer^X algorithm search engine using query "shortest path." The current Citeseer^X algorithm search engine only retrieves pseudo-codes based on textual similarities between user queries and pseudo-codes. This method is effective if the user uses the right keywords. However, there are some cases where the desired algorithms are proposed in different fields of studies, resulting in a different set of context that might not be familiar to the user. The current search engine would treat these algorithms as unrelated to the search query.

4. ALGORITHM CO-CITATION NETWORK

The algorithm co-citation network is an undirected, weighted graph where each node is a document that proposes some algorithms, and each edge weight is the frequency of algorithm co-citation. Formally, for the set of all documents D, the algorithm co-citation network G is defined as follows:

G= <V,E>
V={d | d ∈ D, d proposes one or more algorithms}
E={(a,b) | a,b ∈ V}
Weight((a,b))=|{d | d ∈ D,d cites algorithms in both a and b,(a,b)∈E}|

An algorithm co-citation network is different from a document co-citation network defined in [7] in the sense that each node in the network is a document which proposes algorithms, and one or more of the algorithms are cited in the document. The weight of each link represents the frequency that the algorithms in at least two documents are co-cited. We describe how we detect algorithm citations and construct the algorithm co-citation network in the next sub-sections.

4.1 Algorithm Citation Detection

Given an input document, the algorithm citation detection returns the cited documents whose algorithms are mentioned in the input document. A document is treated as an ordered set of sentences. Briefly, the algorithm citation detector first extracts the set of algorithm-citation sentences in a document. An algorithm citation sentence is defined as a sentence which contains at least one or more algorithm keywords (i.e. 'algorithm', 'method', and 'procedure'), and at least a citation symbol. An example of a sentence where an algorithm is cited is:

> Optimization methods for ECG compression were developed, such as the cardinality constrained shortest path (CCSP) ***algorithm*** presented in [1], [8], [24], [25].

We make an assumption that if both an algorithm keyword and one or more citations appear in a sentence, then it is likely that the citing document mentions the algorithms proposed in the cited documents. After obtaining the set of algorithm-citation sentences, the detector extracts the set of documents cited in such sentences. We use document IDs to represent documents.

4.2 Constructing Algorithm Co-Citation Network

The algorithm co-citation network is constructed by taking the algorithm citations from each document, finding the corresponding document ID for each citation, and creating for each pair of the cited documents an edge of weight 1. If the edge already exists, the weight is incremented by 1. Algorithm 4.1 describes the network construction process. D is the set of input documents from which we extract algorithm citations and construct the algorithm co-citation network. We generate the algorithm co-citation network from roughly 1,370,000 documents in the Citeseer^X repository. The network created contained 9,409,433 edges and roughly 1 million nodes.

Algorithm 4.1: AlgoCoCiteNetworkConstruct(D)

Initialization :
V = {}
E = {}
G = <V,E>

Begin :
  for *each document* d *in* D:
    N ← *list of algorithm-proposing documents cited in* d
    For each (a, b) *where* a, b∈V,a ≠ b:
      if edge (a, b) ∈E:
        Increase weight of edge (a, b) *by* 1
      Else :
        Add edge (a, b) *to* E, *and set the weight to* 1
      End If
  Rerurn G
End.

4.3 Clustering the Algorithm Co-Citation Network

The algorithm co-citation network captures the similarity between two algorithm-proposing documents, based on the assumption that if two algorithms are cited together, then they are likely to be used for similar problems. Such similarity is reflected by the weight of the edge linking the two

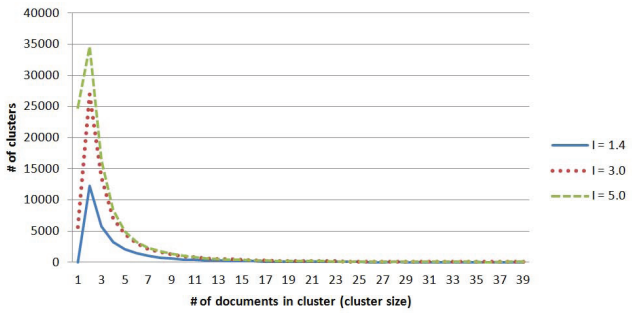

Figure 2: Distribution of cluster sizes with granularity parameters of 1.4, 3.0, and 5.0.

Granularity Level	# clusters	Avg. # documents/cluster
1.4	30,862	14.89
3.0	73,779	6.23
5.0	105,329	4.36

Table 1: Output cluster and average cluster sizes generated using different granularity parameters.

documents in which the co-cited algorithms are mentioned. Based on such a relationship, the network is clustered to produce groups of similar algorithm-proposing documents. A number of clustering tools can be considered; however, the MCL[2] tool worked well for this task. The MCL clustering tool implements the Markov Cluster Algorithm. The algorithm is unsupervised and is based on the simulation of network flow. MCL is designed to specifically cluster large and preferably undirected weighted networks [5].

The other clustering tools that we have considered include Weka[3], Gephi[4], GraphClust[5], and Graclus[6]. Weka [6] supports datasets where all the data points have the same set of attributes. Such data points are different from nodes in the algorithm co-citation network where the only information associated with a node is the similarities between the node and its neighbors. GraphClust is also a network clustering tool; however, it only supports undirected graphs with uniform edge weights. Gephi and Graclus presented problems for large datasets such as ours. For our experiments, they all crashed on an input network of 600,000 edges and 33,601 nodes.

5. EXPERIMENT AND EVALUATION ON CLUSTERING RESULTS

The MCL tool was the most suitable tool we found for clustering large graphs. However, the complete algorithm co-citation network generated from the whole Citeseer^X repository was still too large for the tool to handle. As such, for experimental purposes, we generated a subgraph by randomly selecting 3,000,000 edges from the complete network. The selected subgraph contains 459,585 nodes. Figure 2 shows the distributions of the cluster sizes with different

[2]http://micans.org/mcl/man/mcl.html
[3]http://www.cs.waikato.ac.nz/ml/weka/
[4]http://gephi.org/
[5]http://cs.nyu.edu/shasha/papers/GraphClust.html
[6]http://www.cs.utexas.edu/users/dml/Software/graclus.html

C#	Keywords	#D	#RD	Pr(%)
1	integration, structural, analysis, dynamic, time	41	29	70.73
2	walking, biped, control, robot, locomotion	60	49	81.67
3	technology, programmable, error, cmos, performance	41	20	48.78
4	geometric, constraint, rigidity, graph, system	45	30	66.67
5	mammogram,detection,microcalcification,digital,clustering	43	30	69.77
6	requirement, diagram, model, object-oriented, statechart	46	23	50.00
7	knowledge, ontology, system, design, management	40	31	77.50
8	neural, parallel, network, mapping, architecture	41	32	78.05
9	scheduling, crew, system, transport, driver, transit	43	34	79.07
10	reduction, eigenvalue, power, dominant, system	46	35	76.09
Avg		**44.6**	**31.3**	**70.73**

Table 2: Precision calculated from the sample clusters using granularity parameter 1.4.

clustering granularity levels. With granularity parameter of 1.4 (I = 1.4), the clustering results tend to be coarse-grained, leading to the smallest number of clusters, but highest average number of documents per cluster. At the other extreme where the granularity parameter is 5.0 (I = 5.0), the clustering results tend to be fine-grained, resulting in the largest number of output clusters, and smallest average number of documents per cluster. The granularity parameter of 3.0 (I = 3.0) results in values somewhere in between. Table 1 lists the numbers of output clusters and average cluster sizes for each granularity parameter.

This experiment aims to measure the meaningfulness of the clustering results. The experiment and evaluation use the following steps:

1. Run the MCL clustering algorithm with granularity parameters of 1.4, 3.0, and 5.0 on the sample network with 3,000,000 edges and 459,585 nodes.
2. Randomly choose 10 clusters from each clustering result.
3. For each cluster:
   (a) Retrieve the paper titles of all the documents in the cluster.
   (b) From all the document titles, run the term frequency count to determine top 5 keywords. We use these keywords to describe the cluster.
   (c) Examine each document (manually) and determine the number of documents in the cluster which are related to these 5 keywords.
   (d) Calculate the precision of the cluster, where the precision is the ratio of the number of related documents to the number of all the documents in the cluster.
4. Calculate the average precision of all the 10 clusters.
5. Compare the average precisions of the results from the three granularity levels.

Table 2, 3, and 4 lists the results of the precision measurement of a sample of 10 clusters selected from each of the clustering results generated with different granularity parameters. **C#** is the cluster number, **#D** is the number of documents in the cluster, **#RD** is the number of documents relevant to the 5 chosen keywords, and **Pr(%)** is the precision in percent.

It is worth noting that the highest average precision is achieved in the clustering result using the granularity parameter of 1.4. This granularity level produces the most coarse-grained clustering among all the three parameters. It is also interesting to see that the average precision tends to decrease as the clustering results are more fine-grained.

C#	Keywords	#D	#RD	Pr(%)
1	reduction, model, eigenvalue, power, system	40	13	32.50
2	element, finite, superconvergence, analysis, recovery	52	46	88.46
3	recovery, distribute, rollback, synthesis, system	54	28	51.85
4	java, program, analysis, compiler, object-oriented	58	38	65.52
5	programming,approximation,problem,algorithm,application	48	30	62.50
6	algorithm, system, stability, matrix, linear	52	42	80.77
7	spectral, system, analysis, estimation, identification	46	25	54.35
8	partition, parallel, architecture, language, dataflow	40	30	75.00
9	distributed, signal, sparse, linear, sensor	49	30	61.22
10	feature, selection, analysis, learn, approach	56	26	46.43
Avg		**49.5**	**30.8**	**61.86**

Table 3: Precision calculated from the sample clusters using granularity parameter 3.0.

C#	Keywords	#D	#RD	Pr(%)
1	clustering, data, algorithm, sampling, spatial	47	30	63.83
2	nonlinear, equation, dynamic, solution, schrodinger	41	21	51.22
3	animation, method, simulation, model, realistic	52	32	61.54
4	convex, version, enumeration, facet, version	48	26	54.17
5	match, recognition, contour, object, shape	56	32	57.14
6	waveguide, microwave, multiplexer, design, analysis	51	31	60.78
7	language, entropy, maximum, model, statistical	55	32	58.18
8	testing, protocol, conformance, generation, machine	41	34	82.93
9	learning,reinforcement,stochastic,markov,approximation	43	22	51.16
10	motion, estimation, image, structure, optical	58	33	56.90
Avg		**49.2**	**29.3**	**59.78**

Table 4: Precision calculated from the sample clusters using granularity parameter 5.0.

6. SUGGESTED APPLICATIONS

The clustering on the algorithm co-citation network provides us with groups of related algorithm-proposing documents. We can create clusters of similar algorithms by grouping together pseudo-codes extracted from the documents in the same clusters. There are a number of applications that can be employed from such algorithm clustering. Here, we propose two possible applications which can potentially be used to improve the Citeseer^X algorithm search engine or any similar full text digital library.

6.1 Algorithm Recommendation

Suppose every algorithm in each cluster addresses similar problems. When the algorithm search engine retrieves an algorithm, the algorithms in the same cluster can be displayed as recommended algorithms.

6.2 Improving Ranking

Assuming that algorithms in the same cluster are similar in the sense that they address similar problems, then when an algorithm is textually matched with the search query, the algorithms in the same cluster could receive extra scores so they can be ranked higher. To do this, we propose a technique similar to query expansion:

1. First, the search engine pre-generates a list of keywords that represent each cluster. The keywords may be chosen from the documents which propose the algorithms in the cluster, or they may be chosen from the information (i.e. captions, reference sentences, and synopses) of the algorithms in the clusters.

2. When a user inputs a query, the query engine modifies the original query such that if the original query contains at least one word that belong to the list of keywords of a cluster, then some or all of the keywords that represents such cluster are added to the original query. Adding cluster keywords to an original query would make sure that when at least an algorithm matches the original search query, ranking scores are also given to the rest of the algorithms in the same cluster.

7. CONCLUSIONS AND FUTURE WORK

The algorithm co-citation network captures the similarity between algorithm-proposing documents. The similarity comes from the observation that if two algorithms are cited together, then it is likely that these algorithms address similar problems. Based on this assumption, clustering the network would result in groups of similar algorithm-proposing documents. The clustering results could be applied in many applications to improve the current Citeseer^X or any algorithm search engine, such as improving the ranking and recommending related algorithms. There are several aspects of the methodology that could be improved. For example content analysis techniques can be used to identify additional features which would infer similarity between two algorithm proposing documents. This will generate a different type of algorithm graph. It would be useful to implement clustering algorithms which can handle large networks such as our dataset making it possible to perform experiments on the complete algorithm co-citation network generated from the whole Citeseer^X repository. Using the methods proposed here, many algorithm recommendation systems should be feasible.

8. ACKNOWLEDGMENTS

We gratefully acknowledge useful suggestions from S. Bhatia and S. Das, plus partial support from the NSF and DTRA.

9. REFERENCES

[1] S. Bhatia, S. Lahiri, and P. Mitra. Generating synopses for document-element search. *Proceeding of the 18th ACM conference on Information and knowledge management - CIKM '09*, page 2003, 2009.

[2] S. Bhatia and P. Mitra. Summarizing Figures , Tables and Algorithms in Scientific Publications to Augment Search Results. *ACM Transactions on Information Systems (TOIS)*, pages 1–24, 2010.

[3] S. Bhatia, P. Mitra, and C. L. Giles. Finding algorithms in scientific articles. *Proceedings of the 19th international conference on World wide web - WWW '10*, page 1061, 2010.

[4] S. Bhatia, S. Tuarob, P. Mitra, and C. L. Giles. An Algorithm Search Engine for Software Developers. *SUITE '11: Proceedings of 2011 ICSE Workshop on Search-driven Development: Users, Infrastructure, Tools and Evaluation, 2011*, 2011.

[5] S. Dongen. Graph clustering by flow simulation [Ph.D. dissertation]. *Centers for Mathematics and Computer Science University of Utrecht*, 2000.

[6] M. Hall, H. National, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA data mining software: an update. *SIGKDD Explorations*, 11(1):10–18, 2009.

[7] H. SMALL. Co-citation in the Scientific Literature : A New Measure of the Relationship Between Two Documents. *Journal of the American Society for Information Science*, pages 265–269, 1973.