# ASSG: Adaptive structural summary for RDF graph data

Haiwei Zhang, Yuanyuan Duan, Xiaojie Yuan, and Ying Zhang*

Department of Computer Science and Information Security, Nankai University.
94,Weijin Road, Tianjin, China
{zhanghaiwei,duanyuanyuan,yuanxiaojie,zhangying}
@dbis.nankai.edu.cn
http://dbis.nankai.edu.cn

**Abstract.** RDF is considered to be an important data model for Semantic Web as a labeled directed graph. Querying in massive RDF graph data is known to be hard. In order to reduce the data size, we present ASSG, an **A**daptive **S**tructural **S**ummary for RDF **G**raph data by bisimulations between nodes. ASSG compresses only the part of the graph related to queries. Thus ASSG contains less nodes and edges than existing work. More importantly, ASSG has the adaptive ability to adjust its structure according to the updating query graphs. Experimental results show that ASSG can reduce graph data with the ratio 85% in average, higher than that of existing work.

**Keywords:** Adaptive structural summary, RDF graph, Equivalence class

## 1 Introduction

The resource description framework (RDF) data model has been designed as a flexible representation of schema-relaxable or even schema-free information for the Semantic Web [1]. RDF can be modeled by a labeled directed graph and querying in RDF data is usually thought to be a process of subgraph matching. The subgraph matching problem is defined as follows: for a data graph $G$ and a query graph $Q$, retrieve all subgraphs of $G$ that are isomorphic to $Q$. Existing two solutions, subgraph isomorphism and graph simulation, are expensive where subgraph isomorphism is NP-complete and graph simulation takes quadratic time. Further, indices are used to accelerate subgraph queries on large graph data, but indices incur extra cost on construction and maintainence (see [2] for a survey). Motivated by this, a new approach, using *graph compression*, has been proposed recently [3]. In [3], Fan et al. proposed query preserving graph compression $G_r$, which compresses massive graph into a small one by partitioning nodes into equivalence classes. For subgraph matching, $G_r$ can reduce graph data with the ratio 57% in average. However, for a designated query graph, lots of components (nodes and edges) in $G_r$ are redundant. Hence it is possible to construct a compressed graph for designed subgraph matching.

* Corresponding author.

In this paper, we present ASSG (**A**daptive **S**tructural **S**ummary of **G**raphs), a graph compression method that further reduces the size of the graph data. ASSG has less components than $G_r$ and more importantly, it has adaptive ability to adjust its structure according to different subgraph matchings. In the following sections, we mainly introduce our novel technique.

## 2 Adaptive Structural Summary

In this section, we present our approach of adaptive structural summary for labeled directed graph data (such as RDF). ASSG is actually an compressed graph constructed by equivalence classes of nodes and it has adaptive ability to adjust its structure according to different query graphs.

Graph data is divided into different equivalence classes by bisimulation relations as [3] proposed. For computing bisimulation relation, we refer to the notion *rank* proposed in [4] for describing structural feature from leaf nodes (if exist). A.Dovier, et al.[4] proposed function of computing ranks of nodes for both directed acyclic graph (DAG) and directed cyclic graph (DCG). *Rank* is something like structural feature of nodes from leaf nodes in graph data.

An equivalence class $EC_G$ of nodes in graph data $G = (V, E, L)$ is denoted by a triple $(V_e, R_e, L_e)$, where (1) $V_e$ is a set of nodes included in the equivalence class, (2) $R_e$ is the *rank* of the nodes, and (3) $L_e$ denotes the labels of the nodes.

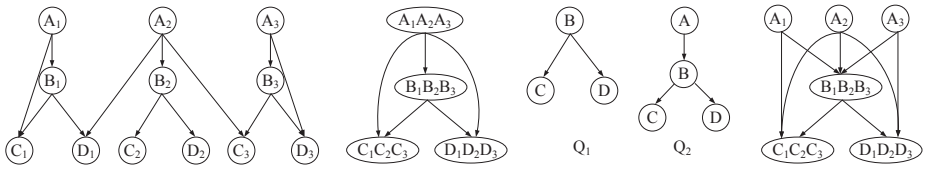


**Fig. 1.** Graph data and equivalence classes

Fig. 1 shows examples of a graph (Fig. 1(a)) and the equivalence classes of nodes (Fig. 1(b)). Labels and ranks of nodes in the same equivalence class are the same, such as $rank(C_1) = rank(C_2) = rank(C_3) = 0$, $rank(A_1) = rank(A_2) = rank(A_3) = 2$, and so on. Two times of DFS processing will be performed to construct equivalence classes of node. DCG will be changed into DAG by algorithm of Tarjan in the first DFS (not shown in Fig. 1). Subsequently, in the second DFS, rank of each node will be measured and then the node will be collapsed into corresponding equivalence class by its label and rank. Hence, $V$ of $G$ is partitioned to different equivalence classes with a cost of $O(|V| + |E|)$.

For a labeled directed graph $G = (V, E, L)$, We define ASSG as $G_{ASS} = (V_{ASS}, E_{ASS}, L_{ASS}, R_{ASS})$, where: (1) $V_{ASS}$ denotes the set of nodes that collapsed by the nodes in the same equivalence class, (2) $E_{ASS}$ is the set of edges, (3) $L_{ASS}$ is the labels of nodes in $V_{ASS}$, (4) $R_{ASS}$ records the *rank* of each $v \in V_{ASS}$.

Obviously, ASSG is the minimum pattern that can describe labeled directed graph data because nodes with the same label and rank will be collapsed. Unfortunately, the process of measuring ranks will lose some descendants or ancestors of nodes. And this case will not conform to the definition of bisimulation, and thus bring out wrong answers for subgraph matching. For example, in Fig. 1(b), the nodes $A_1$ and $A_2$ in the same equivalence class have different children. To solve the problem, ASSG will adaptively adjust its structure for updating query graphs.

For each subgraph matching, the procedure of adaptively updating ASSG includes two stages: matching and partitioning. Given a query graph $Q = (V_Q, E_Q, L_Q)$ and ASSG $G_{ASS} = (V_{ASS}, E_{ASS}, L_{ASS}, R_{ASS})$, assuming that $R_Q$=$\{rank(v_Q)|v_Q \in V_Q\}$. For the matching stage, $\forall v \in V_Q$ and $u \in V_Q$, $\exists v'$, $u' \in V_{ASS}$, if $L_Q(v) = L_{ASS}(v')$, $L_Q(u) = L_{ASS}(u')$, and $R_Q(v) - R_Q(u) = R_{ASS}(v') - R_{ASS}(u')$, then $v$, $u$ matches $v'$, $u'$ respectively. For the partitioning stage, nodes in ASSG matching current query graph will be partitioned into different parts according to its neighbors by the algorithm presented in [5] with the complexity of time $O(|E|log|V_Q|)$. In Fig. 1(c), ASSG will not change while matching $Q_1$, but ASSG will change to the structure shown in Fig. 1(d) while matching $Q_2$. It is obvious that the size of ASSG will increase after further partition, but each partition will adjust minimum amount of nodes. While subgraph matching focuses on frequent nodes, ASSG will remain stable.

## 3 Experimental Evaluation

In this section, we performed experiments on both realistic and synthetic data sets to verify the performance of ASSG.

**Table 1.** Compress Ratio of $G_r$ and ASSG

| Data Set | $\|G\| < \|V\|, \|E\|, \|L\| >$ | $G_r$ | ASSG(15%) |
|---|---|---|---|
| California | 60K<24K, 32K, 95> | 49.22% | 33.25% |
| Internet | 530K<96K, 421K, 50> | 42.41% | 17.08% |
| Citation | 1.7M<815K, 806K, 67> | 31.71% | 5.83% |
| Synthetic | 2.6M<1.4M, 2.1M, 60> | 26.9% | 3.73% |

Firstly, we use compression ratio as a measurement for evaluating the effectiveness of ASSG for subgraph matchings compared with $G_r$. We define compression ratio of ASSG as: $C_{ASS} = |V_{ASS}|/|V|$. Similarly, the compression ratio of $G_r$ is $C_{G_r} = |V_r|/|V|$. The ration is lower, the better. The effectiveness of ASSG compared with $G_r$ is reported in Table 1 where $|G|$ denotes to the size of graph data. For a query graph $G_q = (V_q, E_q, L_q)$, the compression ratio of ASSG is decided by the number of labels $|L_q|$ in the query graph. Assuming that $|L_q| = 15\% \times |L|$, then we can study from table 1: By ASSG, graph data can be highly compressed according to query graphs. ASSG reduces graph data by 85% in average. The compression ratio of ASSG is lower than that of $G_r$.

Secondly, we evaluate the efficiency of updating ASSG. Assuming that number of labels in query graph is 15% of $|L|$. We generate two query graphs for

updating ASSG. The number of repeated labels in these two graphs are 0, 1, 2, 5 respectively as table 2 shows. We can study that the more repeated labels in different query graphs, the less time occupation for ASSG to update. As a result, for frequent subgraph matchings, ASSG can be updated and maintained with low cost of time.

**Table 2.** Time Occupations of Updating ASSG (s)

| Data Set | 0 repeated label | 1 repeated label | 2 repeated labels | 5 repeated labels |
|---|---|---|---|---|
| California | 8.95 | 2.96 | 2.79 | 2.73 |
| Internet | 28.64 | 25.42 | 21.29 | 9.9 |
| Citation | 55.49 | 53.7 | 47.1 | 6.35 |
| Synthetic | 113.47 | 101.32 | 91.24 | 33.73 |

## 4 Conclusion and Future work

We have proposed ASSG, adaptive structural summary for RDF graph data. ASSG is based on equivalence classes of nodes, and ASSG compresses graph data according to the query graphs. We presented main idea for constructing and updating ASSG and designed experiments on realistic and synthetic data sets to evaluate the effectiveness and efficiency of our technique. Experimental results show that the compression ratio of ASSG is lower than that of existing work $G_r$ and ASSG is efficiently updated for frequent queries. Further more, we will use ASSG for optimizing SPARQL queries on RDF data for semantic web.

**Acknowledgments.** This work is supported by National Natural Science Foundation of China under Grant No. 61170184, 61402243, the National 863 Project of China under Grant No. 2013AA013204, National Key Technology R&D Program under Grant No.2013BAH01B05, and the Tianjin Municipal Science and Technology Commission under Grant No.13ZCZDGX02200, 13ZCZDGX01098 and 13JCQNJC00100.

## References

1. T.Neumann., G.Weikum.: The rdf-3x engine for scalable management of rdf data. VLDB J., 19(1), 91–113, 2010.
2. Z.Sun., H.Wang., H.Wang., B.Shao., J.Li.: Efficient Subgraph matching on billion node graphs. The VLDB Journal, 5(9), 788–799 (2012)
3. W.Fan., J.Li., X.Wang., Y.Wu.:Query preserving graph compression. In: ACM SIGMOD International Conference on Management of Data, pp. 157–168. ACM, New York (2012)
4. A.Dovier., C.Piazza., A.Policriti.: A fast bisimulation algorithm. In: Conference on Computer Aided Verification, pp. 79–90. Springer-Verlag Berlin Heidelberg (2001)
5. R.Paige., R.E.Tarjan., R.Bonic.: A linear time solution to the single function coarsest partition problem. Theoretical Computer Science, 40(1), 67–84 (1985)