StructSim: Querying Structural Node Similarity at Billion Scale

Xiaoshuang Chen[§], Longbin Lai[♮§], Lu Qin[†], Xuemin Lin[§♮]

[§]*University of New South Wales,* [†]*University of Technology Sydney,* [♮]*East Normal China University*
xiaoshuang.chen@unsw.edu.au, llai@cse.unsw.edu.au, lu.qin@uts.edu.au, lxue@cse.unsw.edu.au

***Abstract*—Structural node similarity is widely used in analyzing complex networks. As one of the structural node similarity metrics, role similarity has the good merit of indicating automorphism (isomorphism). Existing algorithms to compute role similarity (e.g., RoleSim and NED) suffer from severe performance bottlenecks, and thus cannot handle large real-world graphs. In this paper, we propose a new framework StructSim to compute nodes' role similarity. Under this framework, we prove that StructSim is guaranteed to be an admissible role similarity metric based on the maximum matching. While maximum matching is too costly to scale, we then devise the BinCount matching to speed up the computation. BinCount-based StructSim admits a precomputed index to query one single pair in $O(k \log D)$ time, where k is a small user-defined parameter and D is the maximum node degree. Extensive empirical studies show that StructSim is significantly faster than existing works for computing structural node similarities on the real-world graphs, with comparable effectiveness.**

***Index Terms*—Node Similarity, Role Similarity, Efficiency, Scalability.**

I. Introduction

Structural node similarity is an important metric in graph analysis, and hence has attracted many attentions in the academia [3], [4], [10]. Among these efforts, role similarity [4] stands out because of the property of *automorphism confirmation*[1], that the similarity between two nodes u and v is 1 (the maximum similarity value) if there is an automorphism from u to v. In this paper, given an *unlabelled* and *undirected* simple graph, we study the problem of computing nodes' structural similarity with automorphism confirmation, or more specifically, the role similarity. The term "role" literally refers to the *role* that a node plays in the graph. Examples of roles are a scholar's academic rank in the academia, enzymes in the PPI network [1] and authorities in the web graph [6], to name a few. An effective way to infer the role of a node in a graph is to encompass the configuration of its neighbours. In this sense, we can evaluate the role similarity between two nodes by cross-comparing their contexts of neighbourhood.

Applications. Role similarity is adopted in a wide scope of applications. In social science, role similarity can facilitate role discovery [9]. In bioinformatics, role similarity can be used to predict the function of an unknown protein, given that proteins with similar roles in the PPI network have similar functions [1]. In the world trade network, role similarity is useful for predicting the position of a country in the world system [8]. In a set of IP communication graphs of different time sequences, we can perform role analysis in one graph, and leverage role similarity to explore roles in the others [2]. Other applications include classifying or clustering nodes in the graphs, recommending nodes with similar roles, and detecting anomalous nodes.

State-of-the-art. Jin et al. [4] defined the properties an *admissible role similarity metric* should satisfy, in which *automorphism confirmation* is the most important one. The authors first proved that SimRank, a representative structural similarity measure, fails to guarantee the automorphism confirmation. Thus, SimRank is inappropriate to measure role similarity. The authors then proposed RoleSim as the first admissible role similarity metric. The RoleSim algorithm[2] inherits the framework of SimRank to compute the role similarities of *all pairs* of nodes in an iterative way.

Zhu et al. [10] proposed to compute structural node similarity by utilizing the node's *hierarchical structure* - the k-*adjacent tree*. The authors then proposed the NED distance metric between two nodes as the summation of the level-wise *tree edit distance* of the respective k-adjacent trees. Note that NED is originally a distance metric, while it can be trivially turned into a similarity metric via normalization. For consistency, we refer NED as the corresponding similarity metric.

Motivations. RoleSim and NED are the state-of-the-art algorithms to compute role similarity, but both have severe performance bottlenecks, hence only scale to million-scale graphs. RoleSim relies on a maximum matching of the neighbours to guarantee the automorphism confirmation, while it requires $O(kn^2d^2 \log d)$ time to compute the similarities of all pairs of nodes, where k is the number of iterations, n is the number of nodes and d is the average degree. This apparently hinders applying RoleSim to large graphs. In order to speed up the computation, the authors further proposed IcebergRoleSim [5] that only computes node pairs whose similarities are guaranteed to be larger than a given threshold. While this optimization to some extent boosts the algorithm, it does not improve the time complexity in the worst case. Most significantly, RoleSim's iterative framework is constrained to the all-pair computation. One has to pay the all-pair cost even

[1]Note it is isomorphism confirmation while computing similarity between nodes in the different graphs.

[2]In the following, we will use the metric (RoleSim e.g.) and the algorithm to compute the metric interchangeably.

when only interested in ad-hoc queries such as single-pair and single-source queries.

Unlike RoleSim, NED's hierarchical framework does not need the all-pair cost for ad-hoc queries, but it is not scalable either. Specifically, NED pays $O(k\tau^3)$ time to query one single pair, where τ is the average number of nodes in each level of the k-adjacent trees. Note that NED's k-adjacent tree is based on the breadth-first search tree, but it involves the already-visited nodes in multiple levels. As a result, τ often increases exponentially with respect to k. In the experiment, we witness $\tau > 100,000$ for certain nodes in a small graph with only 400 nodes when $k = 3$, and the cubic time complexity makes NED impossible to run.

Contributions. We propose StructSim in this paper to resolve the scalability issues of existing algorithms. Our contributions are summarized as follows.

(1) A new hierarchical framework named StructSim *to compute role similarity.* We propose StructSim that follows the hierarchical framework to better support ad-hoc queries. StructSim is guaranteed to be an admissible role similarity metric based on the maximum matching.

(2) Efficient matching algorithm. We devise the BinCount matching for StructSim to speed up the computation, which empowers StructSim to handle billion-scale graphs.

(3) Extensive experimental studies. We conduct a case study to show that StructSim achieves comparable result quality. We also perform extensive experiments that show StructSim is significantly faster than the existing algorithms. For example, on the Astroph dataset, to answer a single-pair similarity query, StructSim spends less than 1 ms without the index (less than 10 μs with the index), while NED needs more than 300 seconds. For the all-pair similarity computation, StructSim spends less than 60 seconds, while RoleSim cannot terminate within one day.

II. PRELIMINARY

In this paper, we consider the undirected and unlabelled simple graph $G = (V, E)$, where V is the node set and E is the edge set. We denote the number of nodes $|V|$ and the number of edges $|E|$ by n and m respectively. For a node $u \in V$, d_u denotes the degree of node u and $N(u)$ denotes the neighbours of u, i.e., $N(u) = \{v \in V | (u, v) \in E\}$. Specially, d and D denote the average node degree and the maximum node degree respectively. Given two nodes u and v, we denote $\delta(u, v)$ as the *distance* between u and v, namely the length of the shortest path from u to v. Given a subset of the nodes $U \subseteq V$, the *induced graph* $G(U)$ is defined as a subgraph of G formed from the nodes in U and all edges among U, namely $G(U) = (U, E(U))$, where $E(U) = \{(u, v) | u \in U, v \in U \wedge (u, v) \in E\}$. We have the following definitions.

Definition 1. (i-HOP NEIGHBOURS) *The i-hop neighbours of node u, denoted $N_i(u)$, contain all the nodes whose distance to u are of length i ($i \geq 0$), namely, $N_i(u) = \{v \in V | \delta(u, v) = i\}$. Specifically, $N_0(u) = \{u\}$ and $N_1(u) = N(u)$.*

Definition 2. (i-HOP REACHABLE NEIGHBOURS) *The i-hop reachable neighbours of node u, denoted $N_{\leq i}(u)$, contains all the nodes who have a path to u with length no more than i ($i \geq 0$). Clearly, $N_{\leq i}(u) = \bigcup_{j=0}^{i} N_j(u)$.*

Definition 3. (K-NEIGHBOURHOOD SUBGRAPH) *The k-neighbourhood subgraph of node u is the induced subgraph $G(N_{\leq k}(u))$ (or $G_k(u)$ for short).*

Definition 4. (MATCHING OF i-HOP NEIGHBOURS) *We define the **matching** between two sets S_1 and S_2, denoted $M(S_1, S_2)$, as $M(S_1, S_2) = \{(x, y) | x \in S_m \wedge y = f(x) \in S_M\}$. Here, $f : S_m \to S_M$ is an injective function, where S_m is the smaller set and S_M is the larger set of S_1, S_2. Given nodes u and v, we further denote $M_i(u, v)$ as the matching of the respective i-hop neighbours $N_i(u)$ and $N_i(v)$, namely $M_i(u, v) = M(N_i(u), N_i(v))$. Specially, when $i = 0$, $M_0(u, v) = \{(u, v)\}$. In this paper, we denote $M(u, v) = M_1(u, v)$ as the matching of the neighbours.*

Definition 5. (GRAPH ISOMORPHISM AND AUTOMORPHISM) *An isomorphism of graphs $G = (V_G, E_G)$ and $H = (V_H, E_H)$ is a bijective mapping between V_G and V_H, denoted by $\sigma : V_G \to V_H$, such that for any two nodes u and v in G, $(u, v) \in E_G$ iff $(\sigma(u), \sigma(v)) \in E_H$. Specially, automorphism is an isomorphism mapping G to itself.*

Definition 6. (AUTOMORPHIC EQUIVALENCE) *For nodes u and v in a graph G, if there is an automorphism σ of G satisfying $\sigma(u) = v$, we say that u and v are automorphically equivalent, denoted $u \equiv v$.*

As a node similarity metric that is inferred from the structural context, *role similarity metric* [4] has the following five properties.

Definition 7. (ROLE SIMILARITY METRIC PROPERTIES) *For a graph $G = (V, E)$, a similarity metric $\mathsf{Sim}(u, v)$ that measures the role similarity between node u and node v should satisfy:*

P1. *Range:* $\forall u, v \in V, 0 \leq \mathsf{Sim}(u, v) \leq 1$.

P2. *Symmetry:* $\forall u, v \in V, \mathsf{Sim}(u, v) = \mathsf{Sim}(v, u)$.

P3. *Automorphism confirmation: If* $u \equiv v, \mathsf{Sim}(u, v) = 1$, *where $u \equiv v$ denotes that u and v are automorphically equivalent.*

P4. *Transitive similarity:* $\forall w \in V$, $\mathsf{Sim}(u, w) = \mathsf{Sim}(v, w)$ *if* $u \equiv v$.

P5. *Triangle inequality:* $\forall u, v, w \in V$, $\mathsf{Dist}(u, v) \leq \mathsf{Dist}(u, w) + \mathsf{Dist}(v, w)$, *where* $\mathsf{Dist}(u, v) = 1 - \mathsf{Sim}(u, v)$.

According to [4], a similarity measure is an **admissible role similarity metric** if it satisfies all the five properties.

In this paper, we study the problem of computing the admissible role similarity metric (Definition 7) for any pair of nodes on an undirected and unlabelled simple graph.

III. OUR APPROACHES

As both RoleSim and NED are inefficient to compute, we propose the StructSim framework in this paper that bases the role similarity computation on the hierarchical k-neighbourhood subgraphs. In the following, we first overview the framework by showing how k-neighbourhood subgraph is leveraged to compute the similarity. Then, we prove that StructSim is an admissible role similarity metric based on the maximum matching and give the complexity analysis. Finally, we propose BinCount matching to speed up the computation and analyze the complexity of the algorithm.

A. The StructSim Framework

We outline the framework of StructSim in Algorithm 1. First of all, we assign the initial similarity value of the node pair as the degree ratio according to RoleSim in line 1. Then, StructSim computes the similarity of the node pair via the weighted average of level-wise similarities between their respective k-neighbourhood subgraphs. Specifically, we deploy two operations - `Matching` and `Delta` - to dynamically update the similarity value level by level. At each level i, we can get the i-hop neighbours $N_i(u)$ and $N_i(v)$ from their k-neighbourhood subgraphs (line 3). The `Matching` operation (line 10), as the name suggests, aims at finding a matching $M_i(u,v)$ for $N_i(u)$ and $N_i(v)$. Given $M_i(u,v)$, the `Delta` operation (line 11) then computes the delta value of current level $\Delta_i(u,v)$, which is a summation of $f(x,y)$ for each $(x,y) \in M_i(u,v)$, where $f : (x,y) \rightarrow [0,1]$ is a predefined similarity value of the two nodes. Finally, the similarity value is updated as a weighted average between the old value and $\Delta_i(u,v)$ (line 12). The `Matching` and `Delta` are critical operations for StructSim, which can affect both the admissibility and performance of the algorithm.

Algorithm 1: Framework for StructSim Computation

```
   Input  : The graph $G$, a node pair $(u,v)$ and parameter $k$.
   Output : StructSim$(u,v)$.
1  $S \leftarrow \frac{min(d_u,d_v)}{max(d_u,d_v)}$;
2  foreach level $i \in [1,k]$ do
3      Get $i$-hop neighbours $N_i(u)$ and $N_i(v)$ from $G_k(u)$ and $G_k(v)$;
4      if $N_i(u) = \emptyset$ and $N_i(v) = \emptyset$ then
5          break;
6      else
7          if $N_i(u) = \emptyset$ or $N_i(v) = \emptyset$ then
8              $\Delta_i(u,v) \leftarrow 0$;
9          else
10             (Matching) Computing the matching $M_i(u,v)$;
11             (Delta) $\Delta_i(u,v) \leftarrow \frac{\sum_{(x,y)\in M_i(u,v)} f(x,y)}{max(|N_i(u)|,|N_i(v)|)}$;
12         $S \leftarrow (1-\omega_i)S + \omega_i\Delta_i(u,v)$;
13 return $S$.
```

B. Maximum Matching

Our primary goal is to guarantee the admissibility of StructSim according to Definition 7. Inspired by RoleSim, our first attempt is **maximum matching**, and we accordingly call the algorithm StructSim-Max. Specifically, we compute the matching in line 10 that maximizes the delta value in line 11. Here we use the degree ratio as the predefined similarity value, namely $f(x,y) = \frac{min(d_x,d_y)}{max(d_x,d_y)}$. Intuitively, we want to match the nodes that are most similar regarding their degrees.

We denote the maximum matching at the i^{th} level of u, v as $M_i^{max}(u,v)$, and write StructSim using the following equation:

$$\text{StructSim}(u,v) = \sum_{i=0}^{k} \frac{\tilde{\omega_i} \sum_{(x,y)\in M_i^{max}(u,v)} f(x,y)}{max(|N_i(u)|,|N_i(v)|)}, \quad (1)$$

where $f(x,y) = \frac{min(d_x,d_y)}{max(d_x,d_y)}$ and $\tilde{\omega_i} = \prod_{j=i+1}^{k}(1-\omega_j)\omega_i$. Specifically, $\omega_0 = 1$ and $\sum_{i=0}^{k}\tilde{\omega_i} = 1$. We next prove the admissibility of StructSim-Max.

Theorem 1. (ADMISSIBILITY) StructSim *with the maximum matching is an admissible role similarity metric.*

Proof. Denote $\Delta_i(u,v) = \frac{\sum_{(x,y)\in M_i^{max}(u,v)} f(x,y)}{max(|N_i(u)|,|N_i(v)|)}$ in Equation 1, we just need to prove that for $\forall i \in [1,k]$, Δ_i satisfies all the five properties in Definition 7.

Considering two abstract nodes a and b, node a is linked and only linked to all the nodes in $N_i(u)$ and node b is linked and only linked to all the nodes in $N_i(v)$. Note that after linking to the abstract nodes, the degree of nodes in $N_i(u)$ and $N_i(v)$ remains the same as before. By this definition, we know that if $u \equiv v$, $a \equiv b$ holds since there is a one-to-one equivalence between nodes in $N_i(u)$ and $N_i(v)$. Then $\Delta_i(u,v) = \frac{\sum_{(x,y)\in M_i^{max}(u,v)} f(x,y)}{max(|N_i(u)|,|N_i(v)|)} = \text{RoleSim}^1(a,b)$, where RoleSim is with the degree-ratio initialization. As RoleSim has been proved to satisfy all the properties at any iteration [4], Δ_i meets all the properties as well. □

Complexity Analysis. The time complexity of StructSim-Max is $O(nd + |N_i|^2 \log |N_i|)$, where the term nd comes from online constructing k-neighbourhood subgraphs, and $|N_i|^2 \log |N_i|$ term is from computing the maximum matching by the greedy algorithm, in which $|N_i|$ is the average size of the i-hop neighbours. Note that $|N_i| = n$ in the worst case, which clearly hinders its application to large real-world graphs. RoleSim and NED also suffer from such performance bottleneck. With the constraint of the algorithmic structures, there remain few spaces to further improve RoleSim and NED. However, opportunities still present with the flexibility of the StructSim framework in Algorithm 1.

C. BinCount Matching

We further introduce the BinCount matching in order to accelerate the computation, and accordingly call the algorithm StructSim-BC. We next show how we adjust the `Matching` and `Delta` operations in Algorithm 1 to implement the StructSim-BC algorithm.

Matching Operation. Given two nodes u and v and their k-neighbourhood subgraphs, we arrange the matching process in the i^{th} $(1 \le i \le k)$ level as: we group $N_i(u)$ and $N_i(v)$ into b bins according to the degrees, where the $j^{th}(1 \le j \le b)$ bins are denoted as $N_i^j(u)$ and $N_i^j(v)$ respectively, then we compute the matching $M_i(u,v)$ as

$$M_i(u,v) = \bigcup_{j=1}^{b} M(N_i^j(u), N_i^j(v)), \quad (2)$$

where $M(N_i^j(u), N_i^j(v))$ is a matching of the nodes in the j^{th} bins.

We then arrange the bins based on the logarithmic value of the degree. Specifically, given a node $u \in V$ and its i-hop neighbours $N_i(u)$, let $b = 1 + \lfloor \log D \rfloor$, we will accordingly divide $N_i(u)$ into b bins, where the j^{th} bin $N_i^j(u)$ contains the nodes of degree in the range $[2^{j-1}, 2^j - 1]$. Given the bin arrangement, we define the BinCount set for $N_i(u)$ as

$$\mathcal{B}_i(u) = \{bc_i^1, bc_i^2, \dots, bc_i^b\}, \tag{3}$$

where $bc_i^j = |N_i^j(u)|$. Note that the BinCount sets of each level can be trivially computed while constructing the k-neighbourhood subgraphs.

Delta Operation. We look into a proper predefined similarity value of two matched nodes (x, y), namely $f(x, y)$ in line 11 of Algorithm 1. We agree with the maximum matching process that degree ratio is a proper measurement of the predefined similarity of two nodes. Given the two nodes whose degree are within the range $[2^{j-1}, 2^j - 1]$ (i.e. they are in the same bin), it is obvious that the degree ratio is within $(\frac{1}{2}, 1]$, thus we may use any value in between for $f(x, y)$. In this paper, we simply set $f(x, y) = 1$ which already reaches satisfactory results. By doing so, the `Delta` operation is accordingly simplified as

$$\Delta_i(u, v) = \frac{\sum_{j=1}^{b} \min(bc_i^j(u), bc_i^j(v))}{\max(\sum_{j=1}^{b} bc_i^j(u), \sum_{j=1}^{b} bc_i^j(v))}, \tag{4}$$

where $bc_i^j(u) \in \mathcal{B}_i(u)$ and $bc_i^j(v) \in \mathcal{B}_i(v)$.

Complexity Analysis. The complexity of StructSim-BC is $O(k \log D + nd)$, where D is the maximum node degree and the term nd comes from online computing BinCount sets. Note that we can precompute the BinCount sets as index and improve the complexity to $O(k \log D)$ for a single-pair query. An extra benefit of doing this is that we can trivially do the all-pair similarity computation using $O(n^2 k \log D)$ time while occupying only $O(kn \log D)$ space (for index).

IV. Experimental Evaluation

In this section, we compare StructSim with RoleSim and NED in terms of both effectiveness and efficiency. We implement StructSim (e.g., StructSim-Max and StructSim-BC) in C++. The experiments are by default conducted on a machine configured with an Intel(R) Xeon(R) CPU E3-1220 v6 @ 3.00GHz and 64GB memory.

A. *Effectiveness on Real-World Air-Traffic Networks*

We use three real-world air-traffic networks, i.e., Brazil, Europe and USA from [7] to evaluate the effectiveness of the algorithms, in which Brazil contains 131 nodes, 1,038 edges and 4 labels; Europe contains 399 nodes, 5,995 edges and 4 labels; and USA contains 1,190 nodes, 13,599 edges and 4 labels. Table I presents the precision of each algorithm, in which '-' means NED cannot finish the computation in two days. We can see that the StructSim variants perform similarly in all test cases compared to RoleSim, and NED.

TABLE I
Precision on Air-Traffic Networks

Dataset	RoleSim	NED	StructSim-Max	StructSim-BC
Brazil	0.716	0.725	0.713	0.721
Europe	0.526	-	0.510	0.511
USA	0.557	-	0.540	0.538

TABLE II
Running time (s)

Dataset	Query Time				All-Pair Time	
	NED	SS-Max	SS-BC	SS-BC*	RoleSim	SS-BC*
Astroph	336.9	7.0	0.6ms	1.3μs	63171	57.9
Google	49.7	1.1	0.8ms	2.8μs	-	79058.5
Twitter	-	-	10.4	0.98μs	-	-

B. *Efficiency on Three Large Networks*

We perform query processing and all-pair computation on Astroph, Google and Twitter datasets, in which Astroph contains 18K nodes and 198K edges, Google has 875K nodes and 4M edges and Twitter includes 41M nodes and 1202M edges. Table II summarizes the running time of different algorithms regarding the cases of single-pair query and all-pair computation, in which SS is shorted for StructSim, SS-BC* denotes StructSim-BC with precomputed index and '-' denotes the out-of-memory case. We can observe that StructSim variants are significantly faster than the baseline algorithms, and StructSim-BC is the only one that can query on graphs with billions of edges.

V. Conclusion

In this paper, we present a new framework, namely StructSim to compute nodes' role similarity. Different from SimRank-based iterative framework, StructSim follows the hierarchical scheme, which achieves a more efficient role similarity computation demonstrated by the extensive experimental results on the real-world graphs.

Acknowledgements. Xuemin Lin is supported by 2018YFB1003504, 2019DH-0ZX01, NSFC61232006, DP180103096 and DP170101628. Lu Qin is supported by DP160101513.

References

[1] D. Davis, Ö. N. Yaveroğlu, N. Malod-Dognin, A. Stojmirovic, and N. Pržulj. Topology-function conservation in protein–protein interaction networks. *Bioinformatics*, 31(10):1632–1639, 2015.

[2] K. Henderson, B. Gallagher, T. Eliassi-Rad, H. Tong, S. Basu, L. Akoglu, D. Koutra, C. Faloutsos, and L. Li. Rolx: structural role extraction & mining in large graphs. In *SIGKDD*, pages 1231–1239, 2012.

[3] G. Jeh and J. Widom. Simrank: a measure of structural-context similarity. In *SIGKDD*, pages 538–543, 2002.

[4] R. Jin, V. E. Lee, and H. Hong. Axiomatic ranking of network role similarity. In *SIGKDD*, pages 922–930, 2011.

[5] R. Jin, V. E. Lee, and L. Li. Scalable and axiomatic ranking of network role similarity. *TKDD*, 8(1):3, 2014.

[6] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, 1999.

[7] L. F. Ribeiro, P. H. Saverese, and D. R. Figueiredo. struc2vec: Learning node representations from structural identity. In *SIGKDD*, pages 385–394, 2017.

[8] M. A. Serrano and M. Boguná. Topology of the world trade web. *Physical Review E*, 68(1):015101, 2003.

[9] S. Wasserman and K. Faust. *Social network analysis: Methods and applications*, volume 8. Cambridge university press, 1994.

[10] H. Zhu, X. Meng, and G. Kollios. NED: an inter-graph node metric based on edit distance. *PVLDB*, 10(6):697–708, 2017.