A Framework to Quantify Approximate Simulation on Graph Data

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

[§]*University of New South Wales,* [‡]*Alibaba Group,* [†]*University of Technology Sydney,* [♮]*East Normal China University*
{xiaoshuang.chen,boge.liu}@unsw.edu.au, longbin.lailb@alibaba-inc.com, lu.qin@uts.edu.au, lxue@cse.unsw.edu.au

***Abstract*—Simulation and its variants (e.g., bisimulation and degree-preserving simulation) are useful in a wide spectrum of applications. However, all simulation variants are coarse "yes-or-no" indicators that simply confirm or refute whether one node simulates another, which limits the scope and power of their utility. Therefore, it is meaningful to develop a fractional χ-simulation measure to quantify the degree to which one node simulates another by the simulation variant χ. To this end, we first present several properties necessary for a fractional χ-simulation measure. Then, we present FSim_χ, a general fractional χ-simulation computation framework that can be configured to quantify the extent of all χ-simulations. Comprehensive experiments and real-world case studies show the measure to be effective and the computation framework to be efficient.**

I. Introduction

Consider two directed graphs G_1 and G_2 with labeled nodes from the sets V_1 and V_2, respectively. A *simulation* [30] relation $R \subseteq V_1 \times V_2$ is a binary relation over V_1 and V_2. For each node pair (u, v) in R (namely, u is simulated by v), each u's out-neighbor[1] is simulated by one of v's out-neighbors, and the same applies to in-neighbors. An illustration of this concept is shown below.

Example 1. *As shown in Figure 1, node u is simulated by node v_2, as they have the same label, and each u's out-neighbor can be simulated by the same-label out-neighbor of v_2 (u has no in-neighbors). Note that the two hexagonal nodes in $\mathcal{P}$ are simulated by the same hexagonal node in $\mathcal{G}_2$. Similarly, u is simulated by v_3 and v_4. However, u can not be simulated by v_1, as the pentagonal neighbor of u cannot be simulated by any neighbor of v_1.*

The original definition of simulation put forward by Milner in 1971 [32] only considered out-neighbors. But, in 2011, Ma et al. [30] revised the definition to consider in-neighbors, making it capture more topological information. Additionally, different variants of simulation have emerged over the years, each with its own constraint(s). For example, on the basis that R is a simulation relation, *bisimulation* [33] further requires that R^{-1} is also a simulation, where R^{-1} denotes the converse relation of R (i.e., $R^{-1} = \{(v, u) | \forall (u, v) \in R\}$); and *degree-preserving simulation* [40] requires that two neighbors of u cannot be simulated by the same neighbor of v.

Applications. Simulation and its variants are important relations among nodes, and have been adopted in a wide range of applications. For example, simulation and degree-preserving simulation are shown to be effective in graph pattern matching [15], [30], [31], [40], and a node in the data graph is considered to be a potential match for a node in the query graph if it simulates the query node. Bisimulation has been applied to compute RDF graph alignment [11] and graph partition [19], [37], [44]. Generally, two nodes will be aligned or be placed in the same partition if they are in a bisimulation relation. Other applications include data retrieval [34], graph compression [16] and index construction [17], [23], [36], etc.

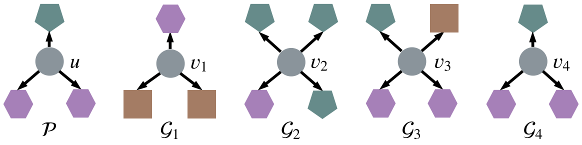


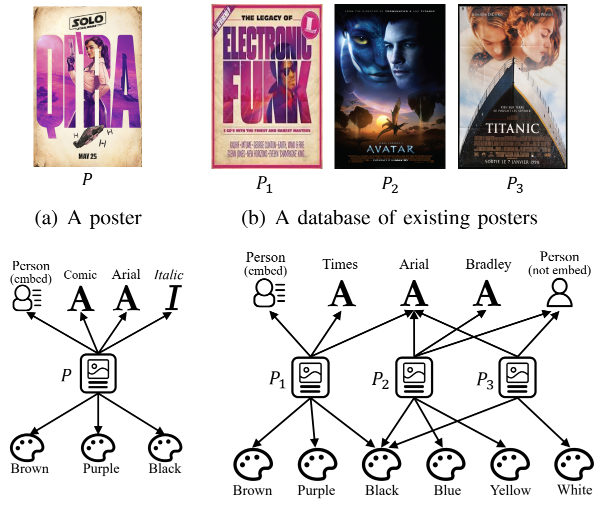
Fig. 1. Example graphs. A node's shape denotes its label.

Motivations. Despite their numerous valuable uses, simulation and its variants are all coarse "yes-or-no" indicators. That is, simulation and its variants can only answer whether node v can fully simulate node u; they cannot tell us whether v might be able to partially or even very nearly (i.e., approximately) simulate u. This coarseness raises two practical issues. First, there often exist some nodes that nearly simulate u in real-world graphs, which either naturally present in the graphs or are consequences of data errors (a common issue by data noise and mistakes of collecting data). However, simulation and its variants cannot catch these nodes and cause loss of potential results. Second, the coarseness makes it inappropriate to apply simulation and its variants to applications that naturally require fine-grained evaluation, such as node similarity measurement. Example 2 provides a real-life illustration of these issues.

Example 2. *We consider the application of simulation to determine whether or not a poster A is simulated by another poster B in terms of their design elements (e.g., color, layout, font, and structure). For example, when compared with the poster P_1 in Figure 2(b), the candidate poster P in Figure 2(a) only slightly differs in the font and font style. Hence, it is highly suspected as a case of plagiarism [7]. Nevertheless, due to a minor change of design elements, there is no exact simulation relation between posters P and P_1, and thus exact simulation can not be used to discover such similarity. As a result, it is more desirable to develop a mechanism to capture the similarity between two posters via the degree of approximate simulation (some fine-grained measurement), instead of simply using the exact simulation.*

Xuemin Lin is the corresponding author

[1] A node u' is an out-neighbor of u, if there is an outgoing edge from u to u' in G. Similarly, u'' is an in-neighbor of u, if an edge from u'' to u presents.



(a) A poster (b) A database of existing posters

(c) Query graph (d) Data graph

Fig. 2. Motivating example. Figures (c) and (d) are graphs representing the posters in (a) and (b), respectively. Nodes are marked with their labels. An edge from nodes u to v indicates that the poster u has a design element v.

In general, it is of practical need to develop a mechanism to quantify the cases of approximate simulation to remedy the impacts of the "yes-or-no" semantics. Such quantification can not only open up a host of possibilities for using simulation but also make the results of simulation more effective and robust. Although the simulation variants differ in certain properties, they are actually derived from a common foundation, namely the simulation relation [32]. Consequently, instead of developing a quantification technique independently and individually for each variant, it is more desirable to devise a general framework that works for all simulation variants. Aside from the obvious benefits of less redundancy, developing a unified framework requires a systematic study of the properties of the different simulation variants. Not only has this not been done before, doing so may help to inspire new variants.

Our Contributions. We propose the *fractional χ-simulation framework* that quantifies the extent of simulation and its variants in the range of $[0, 1]$. Our main contributions are as follows.

(1) A unified definition of χ-simulation. From a systematic study of the properties of simulation and its variants, we distill the base definition of simulation and its variants into a unified definition called χ-simulation. Further, we discover and name a new simulation variant - *bijective simulation*. Theoretically, bijective simulation is akin to the well-known Weisfeiler-Lehman isomorphism test [38] (Section IV-C). Practically, its fractional form (contribution 2) is more effective than the existing models regarding node similarity measurement, as detailed in Section V-D.

(2) A general framework FSim_χ *for computing fractional χ-simulation.* To quantify the degree to which one node simulates another by a χ-simulation, we propose the concept of *fractional χ-simulation* and identify a list of properties that a *fractional χ-simulation* measure should satisfy. Then, we present a general computation framework, namely FSim_χ, which can be configured to compute fractional χ-simulation for all χ-simulations with the properties satisfied. FSim_χ is an iterative framework that computes the fractional χ-simulation scores for all pairs of nodes over two graphs. Furthermore, we show the relations of FSim_χ to several well-known concepts, including node similarity measures (i.e., $\mathsf{SimRank}$ [21] and $\mathsf{RoleSim}$ [22]) and an approximate variant of bisimulation (i.e., k-bisimulation [8], [28], [29], [44]), in Section IV-C.

TABLE I
TABLE OF NOTATIONS

Notation	Description
$G = (V, E, \ell)$	a node-labeled directed graph
$V(G)/E(G)$	the node/edge set of graph G
$\ell(\cdot)$	a labeling function
$N_G^+(u)/N_G^-(u)$	the out-neighbors/in-neighbors of node u in G
$d_G^+(u)/d_G^-(u)$	the out-degree/in-degree of node u in G
d_G	the average degree of G
D_G^+/D_G^-	the maximum out-degree/in-degree of G

(3) Extensive experiments and case studies. We perform empirical studies to exhibit that FSim_χ is robust to parameter tuning and data errors, and is efficient to compute on real-world graphs. We further conduct three case studies to evaluate FSim_χ's potential for subgraph pattern matching, node similarity measurement, and RDF graph alignment. Based on these studies, we reach the following conclusions. First, fractional χ-simulation can remedy the "yes-or-no" semantics of χ-simulation, and it significantly improves the effectiveness of χ-simulation in the related applications, e.g., simulation in subgraph pattern matching. Second, fractional bijective simulation (proposed in this paper) is a highly effective way of measuring node similarity. Finally, the FSim_χ framework provides a flexible way to study the effectiveness of different simulation variants, and thus can be used as a tool to help identify the best variant for a specific application.

II. SIMULATION AND ITS VARIANTS

Data Model. Consider a node-labeled directed graph $G = (V, E, \ell)$, where $V(G)$ and $E(G)$ denote the node set and edge set, respectively (or V and E when the context is clear). Σ is a set of string labels, and $\ell : V \rightarrow \Sigma$ is a labeling function that maps each node u to a label $\ell(u) \in \Sigma$. $N_G^+(u) = \{u' | (u, u') \in E(G)\}$ denotes node u's out-neighbors and, likewise, $N_G^-(u) = \{u' | (u', u) \in E(G)\}$ denotes its in-neighbors. Let $d_G^+(u) = |N_G^+(u)|$ and $d_G^-(u) = |N_G^-(u)|$ be the out- and in-degrees of node u, and let d_G, D_G^+ and D_G^- denote the average degree, maximum out-degree and maximum in-degree of G, respectively. A summary of the notations used throughout this paper appears in Table I.

Simulation Variants. The first step in developing a unified definition of simulation and its variants is to formally define simulation as the foundation of all its variants.

Definition 1. (SIMULATION) *Given the graphs $G_1 = (V_1, E_1, \ell_1)$ and $G_2 = (V_2, E_2, \ell_2)$*[2]*, a binary relation $R \subseteq V_1 \times V_2$ is a simulation if, for $\forall (u, v) \in R$, it satisfies that:*
(1) $\ell_1(u) = \ell_2(v)$,
(2) $\forall u' \in N_{G_1}^+(u), \exists v' \in N_{G_2}^+(v)$ such that (s.t.) $(u', v') \in R$,
(3) $\forall u'' \in N_{G_1}^-(u)$, $\exists v'' \in N_{G_2}^-(v)$ s.t. $(u'', v'') \in R$.

For clarity, u is always a node from V_1, and v is always a node from V_2 in this paper.

[2] $G_1 = G_2$ is allowed in this paper.

The variants of simulation are based on Definition 1 but have additional constraints. Definition 2 below provides a summary of several common simulation variants. However, one exceptional variant, *strong simulation* [30], must be discussed first. Strong simulation is designed for subgraph pattern matching. In brief, strong simulation exists between the query graph Q and data graph G if a subgraph $G[v, \delta_Q]$ of G satisfies the following criteria: (1) a simulation relation R exists between Q and $G[v, \delta_Q]$; and (2) R contains node v and all nodes in Q. Note that the subgraph $G[v, \delta_Q]$ is an induced subgraph that includes all nodes whose shortest distances to v in G are not larger than the diameter δ_Q of Q. In essence, strong simulation essentially performs simulation (Definition 1) multiple times, and so does not need to be specifically defined or further discussed.

Definition 2, which follows, shows how χ-simulation summarizes the base definition of simulation but also considers its variants. The definition below includes two notable ones in *degree-preserving simulation* [40] and *bisimulation* [33].

Definition 2. (χ-SIMULATION) *A simulation relation R by Definition 1 is further a χ-simulation relation, which corresponds to*

- ***Simulation** ($\chi = \mathsf{s}$): no extra constraint;*
- ***Degree-preserving simulation** ($\chi = \mathsf{dp}$): if $(u, v) \in R$, (1) there exists an **injective** function $\lambda_1 : N^+_{G_1}(u) \to N^+_{G_2}(v)$, s.t. $\forall u' \in N^+_{G_1}(u)$, $(u', \lambda_1(u')) \in R$; and (2) there exists an **injective** function $\lambda_2 : N^-_{G_1}(u) \to N^-_{G_2}(v)$, s.t. $\forall u'' \in N^-_{G_1}(u)$, $(u'', \lambda_2(u'')) \in R$;*
- ***Bisimulation** ($\chi = \mathsf{b}$): if $(u, v) \in R$, (1) $\forall v' \in N^+(v)$, $\exists u' \in N^+(u)$ s.t. $(u', v') \in R$; and (2) $\forall v'' \in N^-(v)$, $\exists u'' \in N^-(u)$ s.t. $(u'', v'') \in R$.*

Node u is χ-simulated by node v (or v χ-simulates u), denoted as $u \rightsquigarrow^{\chi} v$, if there is a χ-simulation relation R with $(u, v) \in R$. Specifically, if $u \rightsquigarrow^{\chi} v$ implies $v \rightsquigarrow^{\chi} u$ (i.e., $\chi = \mathsf{b}$), we may use $u \sim^{\chi} v$ directly.

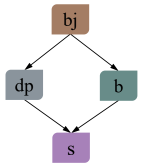
Example 3. *Recall that in Example 1, u is simulated by nodes v_2, v_3 and v_4 in Figure 1. However, u cannot be* dp*-simulated by v_2 because u has two hexagonal neighbors and v_2 does not, which contradicts the requirement of "injective function"; Similarly, u cannot be* b*-simulated by v_3, as v_3's square neighbor fails to simulate any neighbor of u.*

Inspired by the constraints of dp- and b-simulations, we find that a χ-simulation may have the following properties: (1) *injective neighbor mapping* (or IN-mapping for short), i.e., $\forall (u, v) \in R$, two different neighbors (either in or out) of u cannot be mapped to the same neighbor of v; and (2) *converse invariant*, i.e., where $R^{-1} = \{(v, u) | \forall (u, v) \in R\}$ is a χ-simulation if R is a χ-simulation. By Definition 2, dp-simulation has the property of IN-mapping, while b-simulation has converse invariant. The properties of the exiting simulation variants are listed in Figure 3(a).

Remark 1. *Given a χ-simulation with the property of converse invariant, if $u \rightsquigarrow^{\chi} v$, then $v \rightsquigarrow^{\chi} u$ must hold. Therefore, in Definition 2, we have $u \rightsquigarrow^{\mathsf{b}} v$ implies $v \rightsquigarrow^{\mathsf{b}} u$.*

Variants	Properties	
	IN-mapping	Converse Invariant
s-simulation [30]	×	×
dp-simulation [40]	✓	×
b-simulation [33]	×	✓
bj-simulation [this paper]	✓	✓

(a) Properties of each simulation variant (✓ for yes, × for no)



(b) The strictness among χ

Fig. 3. The summarization of all simulation variants.

A New Variant: Bijective Simulation. In compiling Figure 3(a), we realize that no simulation variant had both IN-mapping and converse invariant. This motivated us to define one. Called *bijective simulation*, our definition follows.

Definition 3. (BIJECTIVE SIMULATION) *A simulation relation $R \subseteq V_1 \times V_2$ is a bijective simulation ($\chi = \mathsf{bj}$), if R is a degree-preserving simulation and the functions λ_1 and λ_2, as defined in Definition 2, are further to be surjective (i.e., λ_1 and λ_2 are bijective). Bijective simulation is considered in the χ-simulation (Definition 2) by letting $\chi = \mathsf{bj}$.*

Compared to dp-simulation, bj-simulation requires that the mapping functions of the neighbors to be bijective. In other words, each pair of neighbors in a bj-simulation must be mapped one by one. It's not hard to verify that bj-simulation has the properties of both IN-mapping and converse invariance.

Figure 3(b) shows the strictness among the simulation variants, where a "more-strict" edge from a χ_1- to a χ_2-simulation means that the χ_1-simulation must also be a χ_2-simulation. Such strictness among the variants can also be inferred from Figure 1. More specifically, given $u \rightsquigarrow^{\mathsf{bj}} v_4$, it holds that $u \rightsquigarrow^{\chi} v_4$, $\forall \chi \in \{\mathsf{s}, \mathsf{b}, \mathsf{dp}\}$.

Summary. In this paper, we consider *all together four simulation variants*: simulation ($\chi = \mathsf{s}$), degree-preserving simulation (dp), bisimulation (b), and bijective simulation (bj). With a systematic study of existing simulation variants, we have discovered bijective simulation as a new variant. We believe that our work will further inspire more variants.

Hereafter, we may omit the χ in χ-simulation, referring simply to simulation. To avoid ambiguity, we call the simulation relation in Definition 1 as *simple simulation*.

III. FRACTIONAL SIMULATION

To quantify the degree to which one node simulates the other node, we now set out the properties fractional χ-simulation should satisfy and the framework for its computation.

A. The Properties of Fractional Simulation

Definition 4. (FRACTIONAL χ-SIMULATION) *Given graphs $G_1 = (V_1, E_1, \ell_1)$ and $G_2 = (V_2, E_2, \ell_2)$, and two nodes $u \in V_1$ and $v \in V_2$, the fractional χ-simulation of u and v quantifies the degree to which u is approximately χ-simulated by v, denoted as $\mathsf{FSim}_{\chi}(u, v)$. $\mathsf{FSim}_{\chi}(u, v)$ should satisfy:*

P1. Range: $0 \leq \mathsf{FSim}_{\chi}(u, v) \leq 1$;

*P2. Simulation definiteness: u is χ-simulated by v, i.e., $u \rightsquigarrow^{\chi} v$, **if and only if** $\mathsf{FSim}_{\chi}(u, v) = 1$;*

P3. χ-conditional symmetry: if the χ-simulation has the property of converse invariant (i.e., $u \rightsquigarrow^{\chi} v$ implies $v \rightsquigarrow^{\chi} u$), then $\mathsf{FSim}_{\chi}(u, v)$ should be symmetric, i.e., $\mathsf{FSim}_{\chi}(u, v) = \mathsf{FSim}_{\chi}(v, u)$.

*A computation scheme FSim_{χ} is **well-defined** for fractional χ-simulation, if for $\forall (u, v) \in V_1 \times V_2$, $\mathsf{FSim}_{\chi}(u, v)$ satisfies all three of the above properties.*

Property 1 is a common practice. Property 2 bridges the fractional simulation and the corresponding simulation variant. The sufficient condition reflects the fact that u being χ-simulated by v stands for the maximum degree of their simulation, while the necessary condition (only if) makes fractional simulation imply the case of simulation. Property 3 means the variants with converse invariance (i.e., bisimulation and bijective simulation) can be used as similarity measures.

B. Framework to Compute Fractional Simulation

We propose the FSim_χ framework to compute the fractional χ-simulation scores for all pairs of nodes across two graphs. The FSim_χ is a non-trivial framework because it needs to account for the properties of all simulation variants as well as convergence in general. Note that hereafter, we use FSim_χ interchangeably to indicate the framework and a χ-simulation value.

Recall from Definition 2 that a node u is χ-simulated by node v if they have the same label, and their neighbors are χ-simulated accordingly. Thus, we have divided the computation of $\mathsf{FSim}_\chi(u, v)$ into three parts as follows:

$$\begin{aligned}&\mathsf{FSim}_\chi(u,v) = \\ &\underbrace{w^+ \ \mathsf{FSim}_\chi(N^+_{G_1}(u), N^+_{G_2}(v))}_{\textbf{score by out-neighbors}} + \underbrace{w^- \ \mathsf{FSim}_\chi(N^-_{G_1}(u), N^-_{G_2}(v))}_{\textbf{score by in-neighbors}} \\ &+ \underbrace{(1 - w^+ - w^-) \ \mathcal{L}(u,v)}_{\textbf{score by node label}},\end{aligned} \tag{1}$$

where $\mathsf{FSim}_\chi(N^+_{G_1}(u), N^+_{G_2}(v))$ and $\mathsf{FSim}_\chi(N^-_{G_1}(u), N^-_{G_2}(v))$ denote the scores contributed by the out- and in-neighbors of u and v respectively. w^+ and w^- are weighting factors that satisfy $0 \leq w^+ < 1$, $0 \leq w^- < 1$ and $0 < w^+ + w^- < 1$; and $\mathcal{L}(\cdot)$ is a label function that evaluates the similarity of two nodes' labels. Specifically, if there is no prior knowledge about the labels, $\mathcal{L}(\cdot)$ can be derived by a wide variety of string similarity functions, such as an indicator function, normalized edit distance, Jaro-Winkler similarity, etc. Alternatively, the user could specify/learn the similarities of the label semantics. Since the latter case is beyond the scope of this paper, in the following, we assume no prior knowledge about the labels.

In Equation 1, we need to compute the χ-simulation score between two node sets S_1 and S_2 (the respective neighbors of each node pair). To do so, we derive:

$$\mathsf{FSim}_\chi(S_1, S_2) = \frac{\sum_{(x,y)\in\mathcal{M}_\chi(S_1,S_2)} \mathsf{FSim}_\chi(x,y)}{\Omega_\chi(S_1,S_2)}, \tag{2}$$

where Ω_χ denotes the normalizing operator that returns a positive integer w.r.t. S_1 and S_2. $\mathcal{M}_\chi$ denotes the mapping operator, which returns a set of node pairs defined as:

$$\mathcal{M}_\chi(S_1, S_2; f_\chi) = \{(x, y) \mid x \in X, y = f_\chi(x) \in Y\},$$

where $X \subseteq S_1 \cup S_2$ and $Y \subseteq S_1 \cup S_2$. $f_\chi : X \rightarrow Y$ is a *function* that is subject to certain constraints regarding the simulation variant χ. These constraints include the domain and codomain of f_χ, and the properties that f_χ should satisfy (e.g., that f_χ is an injective function). Note that, for clear presentation, f_χ is always omitted from the mapping operator. How $\mathcal{M}_\chi$ and Ω_χ are configured to deploy different simulation variants for the framework is demonstrated in Section IV.

TABLE II
RESULTS OF WHETHER u IS SIMULATED BY v_i ($i \in \{1, 2, 3, 4\}$) IN FIGURE 1 REGARDING EACH SIMULATION VARIANT (✓ FOR YES, × FOR NO) AND THE CORRESPONDING FRACTIONAL SCORES (IN BRACKET)

Variants	(u, v_1)	(u, v_2)	(u, v_3)	(u, v_4)
s-simulation	× (0.85)	✓ (1.00)	✓ (1.00)	✓ (1.00)
dp-simulation	× (0.72)	× (0.85)	✓ (1.00)	✓ (1.00)
b-simulation	× (0.78)	✓ (1.00)	× (0.93)	✓ (1.00)
bj-simulation	× (0.72)	× (0.81)	× (0.94)	✓ (1.00)

Example 4. *Table II shows the* FSim_χ *scores for some of the node pairs in Figure 1 based on the definition of fractional* χ*-simulation (Definition 4) and the* FSim_χ *framework (Equation 1). We can observe that: (1) a pair* (u, v) *where* u *is not but very closely simulated by* v *has a high* FSim_χ *score, e.g.,* $\mathsf{FSim}_{\mathsf{bj}}(u, v_3)$*; (2) when* u *is* χ*-simulated by* v*,* $\mathsf{FSim}_\chi(u, v)$ *reaches a maximum value of 1, e.g.,* $\mathsf{FSim}_{\mathsf{b}}(u, v_4)$*, which conforms with the well-definiteness of* FSim_χ*.*

According to Equation 1, the FSim_χ score between two nodes depends on the FSim_χ scores of their neighbors. This naturally leads to an iterative computation scheme. This iterative process is detailed in the next section along with how to guarantee its convergence.

C. Iterative Computation

Consider $\mathsf{FSim}^k_\chi(u, v)$, which denotes the χ-simulation score of nodes u and v in the k-[th] iteration ($k \geq 1$), the mapping operator $\mathcal{M}^k_\chi$ and the normalizing operator Ω^k_χ applied in the given iteration.

Initialization. As all simulation variants require an equivalence of node labels (Definition 1 and Definition 2), The FSim_χ score is initially set to $\mathsf{FSim}^0_\chi(u, v) = \mathcal{L}(u, v)$ by default unless otherwise specified. When using such initialization, $\mathcal{L}(u, v) = 1$ must be further constrained if, and only if, $\ell_1(u) = \ell_2(v)$, in order to guarantee that FSim_χ is well-defined (Definition 4).

Iterative Update. According to Equation 1 and Equation 2, the simulation score in the k-[th] iteration for a node pair (u, v) regarding χ is updated via the scores of previous iteration as:

$$\begin{aligned}\mathsf{FSim}^k_\chi(u,v) = &\frac{w^+ \sum_{(x,y)\in\mathcal{M}^k_\chi(N^+_{G_1}(u),N^+_{G_2}(v))} \mathsf{FSim}^{k-1}_\chi(x,y)}{\Omega^k_\chi(N^+_{G_1}(u), N^+_{G_2}(v))} \\ &+ \frac{w^- \sum_{(x,y)\in\mathcal{M}^k_\chi(N^-_{G_1}(u),N^-_{G_2}(v))} \mathsf{FSim}^{k-1}_\chi(x,y)}{\Omega^k_\chi(N^-_{G_1}(u), N^-_{G_2}(v))} \\ &+ (1 - w^+ - w^-)\mathcal{L}(u,v)\end{aligned} \tag{3}$$

Convergence. Below we show what conditions the mapping and normalizing operators should satisfy to guarantee Equation 3 converges. Specifically, the computation is considered to converge if $|\mathsf{FSim}^{k+1}_\chi(u, v) - \mathsf{FSim}^k_\chi(u, v)| < \epsilon$ for $\forall(u, v) \in V_1 \times V_2$, in which ϵ is a small positive value. Note that the simulation subscript χ is omitted in the following theorem as it applies to all simulation variants.

Theorem 1. *The computation in Equation 3 is guaranteed to converge if in every iteration* k*, the following conditions are satisfied for any two node sets* S_1 *and* S_2 *in the mapping and normalizing operators:*

(C1) $|\mathcal{M}^{k+1}(S_1, S_2)| = |\mathcal{M}^k(S_1, S_2)|$*, and* $\Omega^{k+1}(S_1, S_2) = \Omega^k(S_1, S_2)$.

(C2) $|\mathcal{M}^k(S_1, S_2)| \leq \Omega^k(S_1, S_2)$.

(C3) Subject to the function f, $\mathcal{M}^k(S_1, S_2)$ returns node pairs such that

$$\sum_{(x,y)\in\mathcal{M}^k(S_1,S_2)} \mathsf{FSim}^{k-1}(x, y) \textit{ is maximized.}$$

Proof. Let $\delta^k(u, v) = |\mathsf{FSim}^k(u, v) - \mathsf{FSim}^{k-1}(u, v)|$ and $\Delta^k = \max_{(u,v)} \delta^k(u, v)$. To prove this theorem, we must show that Δ^k decreases monotonically, i.e., $\Delta^{k+1} < \Delta^k$.

Let $W^k(S_1, S_2) = \sum_{(x,y)\in\mathcal{M}^k(S_1,S_2)} \mathsf{FSim}^{k-1}(x, y)$. As the size of the mapping operator and the value of normalizing operator between S_1 and S_2 do not vary with k (C1), we simply write $|\mathcal{M}(S_1, S_2)|$ and $\Omega(S_1, S_2)$ by dropping the superscript. Then, we have

$$\begin{aligned} W^{k+1}(S_1, S_2) &\geq \sum_{(x,y)\in\mathcal{M}^k(S_1,S_2)} \mathsf{FSim}^k(x, y) \text{ (by C3)} \\ &\geq W^k(S_1, S_2) - |\mathcal{M}(S_1, S_2)|\Delta^k \text{ (by C1)} \end{aligned}$$

Similarly, $W^k(S_1, S_2) \geq W^{k+1}(S_1, S_2) - |\mathcal{M}(S_1, S_2)|\Delta^k$ can be derived, and we immediately have,

$$|W^{k+1}(S_1, S_2) - W^k(S_1, S_2)| \leq \Omega(N_1, N_2)\Delta^k \text{ (by C2)} \qquad (4)$$

Then,

$$\begin{aligned} \delta^{k+1}(u, v) &\leq (w^+ + w^-)\Delta^k \text{ (by Equation 4)} \\ &< \Delta^k \text{ (by } w^+ + w^- < 1) \end{aligned} \qquad (5)$$

Thus, $\Delta^{k+1} < \Delta^k$, and the computation converges. □

Corollary 1. *The computation in Equation 3 converges within $\lceil \log_{(w^+ + w^-)} \epsilon \rceil$ iterations.*

Proof. According to Equation 5, we have $\Delta^{k+1} \leq (w^+ + w^-)\Delta^k$. As Δ^0 cannot exceed 1, the theorem holds. □

We discuss the needs of the three conditions in Theorem 1. Given node sets S_1 and S_2, C1 requires that the value of the normalizing operator $\Omega_\chi(S_1, S_2)$ and the number of node pairs in $\mathcal{M}_\chi(S_1, S_2)$ (i.e., $|\mathcal{M}_\chi(S_1, S_2)|$) remain unchanged throughout the iterations. C2 requires that $|\mathcal{M}_\chi(S_1, S_2))|$ should be less than $\Omega_\chi(S_1, S_2)$ to guarantee the range property in Definition 4. C3 requires that $\mathcal{M}_\chi$ should include the pairs of neighbors that maximize the sum of their FSim_χ scores in previous iteration. Intuitively, C3 maximizes the contributions of neighbors and is essential to satisfy simulation definiteness (property 2 in Definition 4). Such a mapping operator is accordingly called a *maximum mapping operator*, and will be applied by default in the following.

D. Computation Algorithm

Algorithm 1 outlines the process for computing FSim_χ. The computation begins by initializing a hash map H_c to maintain the initial FSim_χ scores of candidate node pairs (Line 1). Note that not all $|V_1| \times |V_2|$ node pairs need to be maintained, which is explained in Section IV. Then, the scores of the node pairs in H_c are updated iteratively until convergence (Lines 3-10). In Line 11, the hash map is returned with the results .

Parallelization. The most time-consuming part of Algorithm 1 is running the iterative update in Lines 3 through 10. This motivated us to consider accelerating the computation with parallelization by using multiple threads to compute different node pairs simultaneously. In this implementation, the simulation scores of the previous iteration are maintained in H_p, which means computing the node pairs in Lines 7 and 9 is independent of each other, and can be completed in parallel without any conflicts. We simply round-robin the node pairs in H_c to distribute the load to all available threads, which achieves satisfactory scalability in the experiment (Figure 9(a)).

Algorithm 1: The algorithm of computing FSim_χ

```
    Input  : Graphs $G_1 = (V_1, E_1, \ell_1)$, $G_2 = (V_2, E_2, \ell_2)$,
             weighting factors $w^+, w^-$.
    Output : $\mathsf{FSim}_\chi$ Scores.
 1  $H_\mathsf{c} \leftarrow$ Initializing($G_1$, $G_2$, $w^+$, $w^-$);
 2  $H_\mathsf{p} \leftarrow H_\mathsf{c}$;
 3  while not converged do
 4      foreach $(u, v) \in H_\mathsf{c}$ do
 5          $H_\mathsf{c}[(u, v)] \leftarrow (1 - w^+ - w^-)\mathcal{L}(u, v)$;
 6          foreach $(x, y) \in \mathcal{M}_\chi(N^+_{G_1}(u), N^+_{G_2}(v))$ do
 7              $H_\mathsf{c}[(u, v)] \leftarrow H_\mathsf{c}[(u, v)] + \frac{w^+ H_\mathsf{p}[(x,y)]}{\Omega_\chi(N^+_{G_1}(u), N^+_{G_2}(v))}$;
 8          foreach $(x', y') \in \mathcal{M}_\chi(N^-_{G_1}(u), N^-_{G_2}(v))$ do
 9              $H_\mathsf{c}[(u, v)] \leftarrow H_\mathsf{c}[(u, v)] + \frac{w^- H_\mathsf{p}[(x',y')]}{\Omega_\chi(N^-_{G_1}(u), N^-_{G_2}(v))}$;
10      $H_\mathsf{p} \leftarrow H_\mathsf{c}$;
11  return $H_\mathsf{c}$.
```

Upper-Bound Updating. According to the range property (Definition 4) and the computation in Equation 3, there exists an upper-bound on the FSim_χ value of each node pair, which is computed via:

$$\begin{aligned} \mathsf{FSim}_\chi(u, v) &\leq \overline{\mathsf{FSim}}_\chi(u, v) \\ &= \lambda^+(u, v) + \lambda^-(u, v) + (1 - w^+ - w^-)\mathcal{L}(u, v), \end{aligned} \qquad (6)$$

where $\lambda^\mathsf{s} = \frac{w^\mathsf{s}|\mathcal{M}_\chi(N^\mathsf{s}_{G_1}(u), N^\mathsf{s}_{G_2}(v))|}{\Omega^\mathsf{s}_\chi(N^\mathsf{s}_{G_1}(u), N^\mathsf{s}_{G_2}(v))}$, for $\mathsf{s} \in \{+, -\}$. Accordingly, if the upper bound of a certain node pair (u, v) is relatively small (smaller than a given threshold β), it is expected to make a limited contribution to the scores of others. Thus, we can skip computing (and maintaining) $\mathsf{FSim}_\chi(u, v)$, and use an approximated value $\alpha\overline{\mathsf{FSim}}_\chi(u, v)$ ($0 < \alpha < 1$ is a given small constant) instead when needed. The implementation of upper-bound updating based on Algorithm 1 is as follows: (1) in Line 1, H_c only maintains the node pairs that are guaranteed to be larger than β; (2) in Lines 7 and 9, if (x, y) (or (x', y')) is not in H_p, use $\alpha\overline{\mathsf{FSim}}_\chi(x, y)$ (or $\alpha\overline{\mathsf{FSim}}_\chi(x', y')$) instead.

IV. Configure Framework to Quantify Different Simulation Variants

In this section, we show how to configure the mapping and normalizing operators in Equation 2, such that the computation of FSim_χ converges, and FSim_χ remains well-defined (Definition 4) for all simulation variants.

A. Configurations of Simple Simulation

Fractional s-simulation. Given two node sets S_1 and S_2, $\mathcal{M}_\mathsf{s}$ and Ω_s are the operators for implementing fractional s-simulation according to Definition 4 as follows:

$$\mathcal{M}_\mathsf{s}(S_1, S_2) = \{(x, y) | \forall x \in S_1, y = f_\mathsf{s}(x) \in S_2\}, \qquad (7)$$

where $f_{\mathsf{s}} : S_1 \rightarrow S_2$ is a function subject to the label constraint $\mathcal{L}(x, f_{\mathsf{s}}(x)) \geq \theta$, and

$$\Omega_{\mathsf{s}}(S_1, S_2) = |S_1|. \tag{8}$$

Remark 2. ***Label-constrained Mapping.*** *Analogous to the initialization of* FSim_χ *(Section III-C), a label constraint is added when mapping neighbors.* θ *is a constant given by the user to control the strictness of the mapping. When* $\theta = 0$*, the nodes can be mapped arbitrarily. When* $\theta = 1$*, only nodes of the same label can be mapped. It is obvious that a larger* θ *leads to faster computation. As a practical guide to setting* θ*, Section V-B includes a sensitivity analysis of* θ *and Section V-C provides an efficiency analysis. In the following, the label constraint is applied in the mapping operator by default and is thus omitted from the notations for clear presentation.*

Convergence. It is obvious that $|M_{\mathsf{s}}(S_1, S_2)| \leq |S_1| = \Omega_{\mathsf{s}}(S_1, S_2)$, which satisfies C1 and C2 in Theorem 1. As mentioned earlier, C3 is applied by default. Therefore, the convergence of $\mathsf{FSim}_{\mathsf{s}}$ is guaranteed.

Well-Definiteness. Theorem 2 shows that $\mathsf{FSim}_{\mathsf{s}}$ is well-defined for fractional s-simulation according to Definition 4.

Theorem 2. $\mathsf{FSim}_{\mathsf{s}}$ *is well-defined for fractional* s*-simulation.*

Proof. We prove that $\mathsf{FSim}_{\mathsf{s}}$ satisfies all the properties in Definition 4. P1 is easy to verify. It is unnecessary to verify P3 as s-simulation has no converse invariant. Below we prove P2. For brevity, we only consider out-neighbors in the proof, and the proof with in-neighbors is similar.

We first prove that if $\mathsf{FSim}_{\mathsf{s}}(u, v) = 1$, $u \rightsquigarrow^{\mathsf{s}} v$. Based on Equation 1, we have $\ell_1(u) = \ell_2(v)$, and we add (u, v) into R (initialized as $\emptyset$). $\forall (x, y) \in \mathcal{M}_{\mathsf{s}}$, $\mathsf{FSim}_{\mathsf{s}}(x, y) = 1$ and $\ell_1(x) = \ell_2(y)$. Then, we add these nodes pairs into R, i.e. $R = R \bigcup \mathcal{M}_{\mathsf{s}}$. New node pairs can be added recursively. The process will terminate as $|R|$ increases and cannot exceed $|V_1| \times |V_2|$. One can verify that R is a simulation.

We next prove that, for $\forall (u, v) \in R$, $\mathsf{FSim}^k_{\mathsf{s}}(u, v) = 1$ for any k, where R is a simulation relation. Based on Definition 1, we define the mapping $\mathcal{M}$ between $N^+_{G_1}(u)$ and $N^+_{G_2}(v)$ as $\mathcal{M} = \{(u', v') | \forall u' \in N^+_{G_1}(u), (u', v') \in R\}$. The case of $k = 0$ is easy to verify. Assume the theorem holds at $k - 1$. For a node pair $(u, v) \in R$, any $(u', v') \in \mathcal{M}$ defined above satisfies $\mathsf{FSim}^{k-1}_{\mathsf{s}}(u', v') = 1$. Clearly, $\mathcal{M}$ is a mapping operator defined in Equation 7. Thus, $\mathsf{FSim}^k_{\mathsf{s}}(u, v) = 1$. □

Computation. The mapping operator $\mathcal{M}_{\mathsf{s}}$ (Equation 7) constrains that $\forall (x, y) \in \mathcal{M}_{\mathsf{s}}$, $\mathcal{L}(x, y) \geq \theta$. As a result, the nodes pairs with $\mathcal{L}(\cdot) < \theta$ will never contribute to the computation. Thus, only the node pairs with $\mathcal{L}(\cdot) \geq \theta$ need to be maintained(Line 1 in Algorithm 1), which helps to reduce both the time and space complexity.

Cost Analysis. The time cost to compute $\mathsf{FSim}_{\mathsf{s}}(u, v)$ is dominated by the mapping operator. According to Equation 7, for $\forall x \in S_1$, we simply search $y \in S_2$ to maximize $\mathsf{FSim}^{k-1}_\chi(x, y)$, which takes $O(|S_1||S_2|)$ time. Therefore, the time complexity of computing $\mathsf{FSim}_{\mathsf{s}}$ is $O(k|H|(D^+_{G_1}D^+_{G_2} + D^-_{G_1}D^-_{G_2})$ with $|H| \leq |V_1| \times |V_2|$ and k as the number of iterations. The space cost is $O(|H|)$, as the map of $\mathsf{FSim}_{\mathsf{s}}$ scores for the previous iteration needs to be stored.

B. Configurations for All Simulation Variants

Table III summarizes the configurations of each simulation variant. With the given configurations, FSim_χ is well-defined for $\forall \chi \in \{\mathsf{s}, \mathsf{dp}, \mathsf{b}, \mathsf{bj}\}$. We only provide the proofs of the well-definiteness for $\mathsf{FSim}_{\mathsf{s}}$ (asymmetric, Theorem 2) and $\mathsf{FSim}_{\mathsf{bj}}$ (symmetric, Theorem 3). The proofs for the other variants are similar and thus are omitted due to space limitations.

Theorem 3. $\mathsf{FSim}_{\mathsf{bj}}$ *is well-defined for fractional* bj*-simulation.*

Proof. Proofs of P1 and P2 are similar to those of $\mathsf{FSim}_{\mathsf{s}}$ (Theorem 2). Proof of P3, i.e., $\mathsf{FSim}_{\mathsf{bj}}(u, v) = \mathsf{FSim}_{\mathsf{bj}}(v, u)$, is given by mathematical induction.

As the initialization function is symmetric, we have $\mathsf{FSim}^0_{\mathsf{bj}}(u, v) = \mathsf{FSim}^0_{\mathsf{bj}}(v, u)$. Suppose that $\mathsf{FSim}^{k-1}_{\mathsf{bj}}(u, v)$ is symmetric, the symmetry of $\mathsf{FSim}^k_{\mathsf{bj}}(u, v)$ can be immediately proved as Ω_{bj} is symmetric as well. As a result, we have $\mathsf{FSim}_{\mathsf{bj}}(u, v) = \mathsf{FSim}_{\mathsf{bj}}(v, u)$. □

Cost Analysis. According to Algorithm 1, the space complexity is $O(|H|)$, where $|H| \leq |V_1| \times |V_2|$. The time complexity of computing $\mathsf{FSim}_{\mathsf{b}}$ is the same as $\mathsf{FSim}_{\mathsf{s}}$. For computing $\mathsf{FSim}_{\mathsf{dp}}$ and $\mathsf{FSim}_{\mathsf{bj}}$, the Hungarian algorithm needs to be applied to implement the mapping operators due to the presence of injection. Using a popular greedy approximate of Hungarian [9], $\mathcal{M}_{\mathsf{dp}}(S_1, S_2)$ and $\mathcal{M}_{\mathsf{bj}}(S_1, S_2)$ can be solved in a time complexity of $O(|S_1||S_2| \log(|S_1||S_2|))$. As a whole, the time cost of computing $\mathsf{FSim}_{\mathsf{dp}}$ and $\mathsf{FSim}_{\mathsf{bj}}$ is $O(k|H|(D^+_{G_1}D^+_{G_2} \cdot \log D^+_{G_1}D^+_{G_2} + D^-_{G_1}D^-_{G_2} \cdot \log D^-_{G_1}D^-_{G_2}))$.

C. Discussions

FSim_χ is closely related to several well-known concepts, including node similarity measures (i.e., SimRank and RoleSim), k-bisimulation (a variant of bisimulation) and graph isomorphism. In this subsection, we discuss their relations to FSim_χ.

Relations to Similarity Measures. The FSim_χ framework (Equation 3) can be configured to compute SimRank [21] and Rolesim [22]. As both the algorithms are applied to a single unlabeled graph, we let $G_1 = G_2$, and the graph be label-free.

To configure FSim_χ for SimRank, if $u = v$, we set $\mathsf{FSim}^0_\chi(u, v)$ to 1 in the initialization step, and 0 otherwise. In the update step, we set $w^+ = 0$, $\mathcal{M}(S_1, S_2) = S_1 \times S_2$, $\Omega(S_1, S_2) = |S_1||S_2|$ and $\mathcal{L}(u, v) = 0$ in Equation 3. It is clear that with such configurations, FSim_χ computes SimRank scores for all node pairs in a manner following [21]. Note that the convergence of FSim_χ is guaranteed, as the mapping and normalizing operators satisfy all conditions in Theorem 1.

RoleSim [22] computes structural similarity with automorphic confirmation (i.e. the similarity of two isomorphic nodes is 1) on an undirected graph. Thus, we let the out-neighbors of each node maintain its undirected neighbors, and leave the in-neighbors empty. In the initialization step, we set $\mathsf{FSim}^0_\chi(u, v) = \frac{\min(d^+(u), d^+(v))}{\max(d^+(u), d^+(v))}$ for all node pairs following [22]. In the update step, we set $w^- = 0$ and $\mathcal{L}(u, v) = 1$ for each node pair, and follow the settings of mapping and normalizing operators of bijective simulation in Equation 3. With such configurations, one can verify according to [22] that FSim_χ is computing axiomatic role similarity.

TABLE III

CONFIGURATIONS OF THE FRACTIONAL χ-SIMULATION FRAMEWORK (FSim_χ) TO QUANTIFY THE STUDIED SIMULATION VARIANTS.

FSim_χ	$\Omega_\chi(S_1, S_2)$	$\mathcal{M}_\chi(S_1, S_2)$	**Function Constraints (label constraint implied)**
FSim_s	$\|S_1\|$	$\{(x,y) \| \forall x \in S_1, y = f_\mathsf{s}(x) \in S_2\}$	$f_\mathsf{s}(x) : S_1 \to S_2$
$\mathsf{FSim}_\mathsf{dp}$	$\|S_1\|$	$\{(x,y) \| \forall x \in S_1', y = f_\mathsf{dp}(x) \in S_2\}$, where $S_1' \subseteq S_1$ with $\|S_1'\| = \min(\|S_1\|, \|S_2\|)$	$f_\mathsf{dp} : S_1' \to S_2$ is an injective function
FSim_b	$\|S_1\| + \|S_2\|$	$\{(x,y) \| \forall x \in S, y = f_\mathsf{b}(x) \in S\}$, where $S = S_1 \cup S_2$	$f_\mathsf{b}(x) \in \begin{cases} S_2, & \text{if } x \in S_1, \\ S_1, & \text{if } x \in S_2 \end{cases}$
$\mathsf{FSim}_\mathsf{bj}$	$\sqrt{\|S_1\| \times \|S_2\|}$	$\{(x,y) \| \forall x \in S_m, y = f_\mathsf{bj}(x) \in S_M\}$, in which if $\|S_1\| \leq \|S_2\|$, $S_m = S_1$ and $S_M = S_2$; otherwise, $S_m = S_2$ and $S_M = S_1$	$f_\mathsf{bj}(x) : S_m \to S_M$ is an injective function

Relation to k-bisimulation. k-bisimulation [8], [28], [29], [44] is a type of approximate bisimulation. Given a graph $G(V, E, \ell)$ and an integer $k \geq 0$, node u is simulated by node v via k-bisimulation [28] (i.e., u and v are k-bisimilar) if, and only if, the following conditions hold: (1) $\ell(u) = \ell(v)$; (2) if $k > 0$, for $\forall u' \in N_G^+(u)$, there exists $v' \in N_G^+(v)$ s.t. u' and v' are [k-1]-bisimilar; and (3) if $k > 0$, for $\forall v' \in N_G^+(v)$, there exists $u' \in N_G^+(u)$ s.t. v' and u' are [k-1]-bisimilar. An iterative framework is proposed by [28] to compute k-bisimulation, in which each node u is assigned with a signature $sig_k(u)$ based on its node label and neighbors' signatures. Node u is simulated by node v via k-bisimulation if and only if $sig_k(u) = sig_k(v)$ [28]. We show in Theorem 4 that our FSim_χ can be configured to compute k-bisimulation. As k-bisimulation in [28] uses one single graph and only considers out-neighbors, we set $G_1 = G_2$ and $w^- = 0$ for FSim_χ. Recall that $\mathsf{FSim}_\mathsf{b}^k(u, v)$, computed by Equation 3, is the b-simulation score of nodes u and v in the k-[th] iteration,

Theorem 4. *Given graph G and integer k, node u is simulated by node v via k-bisimulation if and only if* $\mathsf{FSim}_\mathsf{b}^k(u, v) = 1$.

Proof. The case when $k = 0$ is easy to verify. Assume the theorem is true at $k - 1$, we show that the theorem also holds at k. On the one hand, if u is simulated by v via k-bisimulation, i.e., $sig_k(u) = sig_k(v)$, one can verify that $\mathcal{M} = \{(u', v') | sig_{k-1}(u') = sig_{k-1}(v') \wedge v' \in N_G^+(v), \forall u' \in N_G^+(u)\} \bigcup \{(v'', u'') | sig_{k-1}(v'') = sig_{k-1}(u'') \wedge u'' \in N_G^+(u), \forall v'' \in N_G^+(v)\}$ is a matching of FSim_b. Based on the assumption, we have $\mathsf{FSim}_\mathsf{b}^k(u, v) = 1$. On the other hand, if $\mathsf{FSim}_\mathsf{b}^k(u, v) = 1$, for $\forall u' \in N_G^+(u)$, there exists $v' \in N_G^+(v)$ such that $\mathsf{FSim}_\mathsf{b}^{k-1}(u', v') = 1$, which means $sig_{k-1}(u') = sig_{k-1}(v')$. Similarly, $\forall v'' \in N_G^+(v)$, there exists $u'' \in N_G^+(u)$ with $sig_{k-1}(u'') = sig_{k-1}(v'')$. Thus, the set of signature values in u's neighborhood is the same as that in v's neighborhood. Then, we have $sig_k(u) = sig_k(v)$. □

Relation to isomorphism. The graph isomorphism test asks for whether two graphs are topologically identical, and node u of G_1 is *isomorphic* to node v of G_2 if there exists an isomorphism between G_1 and G_2 mapping u to v. Graph isomorphism is a challenging problem, and there is no polynomial-time solution yet [10]. The Weisfeiler-Lehman isomorphism test (the WL test) [38] is a widely used solution to test whether two graphs are isomorphic. The WL test can be solved in polynomial time, but it is necessary but not sufficient for isomorphism, that is two graphs that are isomorphic must pass the WL test but not vice versa. Like the WL test, bijective simulation is also necessary but not sufficient for isomorphism. We next show that it is as powerful as the WL test in theory.

The WL test [38] is applied to undirected labeled graphs, and the graph model is accordingly adapted as $\mathsf{RoleSim}$. We assume both graphs are connected, as otherwise each pair of connected components can be independently tested. Given graphs G_1 and G_2, the WL test iteratively labels each node $u \in V_1$ (resp. $v \in V_2$) as $s(u)$ (resp. $s(v)$). The algorithm decides that node u is isomorphic to node v if $s(u) = s(v)$ when the algorithm converges[3] The following theorem reveals the connection between WL test and bijective simulation.

Theorem 5. *Given graphs G_1 and G_2, and a node pair $(u, v) \in V_1 \times V_2$, and assume the WL test converges, we have $s(u) = s(v)$* ***if and only if*** $\mathsf{FSim}_\mathsf{bj}(u, v) = 1$, *namely* $u \sim^{\mathsf{bj}} v$.

Proof. Let $s^k(u)$ and $s^k(v)$ be the label of u and v at the k-[th] iteration during WL test. We first prove that for any k, if $s^k(u) = s^k(v)$, $\mathsf{FSim}_\mathsf{bj}^k(u, v) = 1$. The case of $k = 0$ is easy to verify. Suppose the theorem is true at $k - 1$. At the k-[th] iteration, we have $s^k(u) = s^{k-1}(u) \sqcup_{u' \in N(u)} s^{k-1}(u')$ and $s^k(v) = s^{k-1}(v) \sqcup_{v' \in N(v)} s^{k-1}(v')$, where $\sqcup$ denotes label concatenation. If $s^k(u) = s^k(v)$, there exists a bijective function $\lambda_1 : N_{G_1}(u) \to N_{G_2}(v)$ s.t. $s^{k-1}(u') = s^{k-1}(\lambda_1(u'))$. Based on the assumption, we have $\mathsf{FSim}_\mathsf{bj}^k(u, v) = 1$.

Next, we prove if $u \sim^{\mathsf{bj}} v$, $s(u) = s(v)$. It is easy to verify the case of $k = 0$. Assume that if $\mathsf{FSim}_\mathsf{bj}^{k-1}(u, v) = 1, s^{k-1}(u) = s^{k-1}(v)$ holds. At the k-[th] iteration, if $\mathsf{FSim}_\mathsf{bj}^k(u, v) = 1$, there exits a bijective function $\lambda_2 : N_{G_1}(u) \to N_{G_2}(v)$ s.t. for $\forall u' \in N_{G_1}(u)$, $\mathsf{FSim}_\mathsf{bj}^{k-1}(u', \lambda_2(u')) = 1$. Thus, we can derive $s^{k-1}(u') = s^{k-1}(\lambda_2(u'))$ and $s^k(u) = s^k(v)$. □

Remark 3. *Note that there is no clear relation between bijective simulation and graph homomorphism. To be specific, bijective simulation cannot derive homomorphism, and homomorphism cannot derive bijective simulation either.*

V. EXPERIMENTAL EVALUATION

A. Setup

Datasets. We used eight publicly available real-world datasets. Table IV provides their descriptive statistics, including the number of nodes $|V|$, the number of edges $|E|$, the number of labels $|\Sigma|$, the average degree d_G, the maximum out-degree D_G^+ and the maximum in-degree D_G^-.

Experimental Settings. Without loss of generality, we assume that in-neighbors and out-neighbors contribute equally to the FSim_χ computation. Thus, $w^+ = w^-$ in all experiments. Algorithms were terminated when the values changed by less than 0.01 of their previous values. *Note that when we applied* FSim_χ *to one single graph, we actually computed the* FSim_χ

[3] The algorithm is not guaranteed to converge.

TABLE IV
DATASET STATISTICS AND SOURCES

Datasets	$\|E\|$	$\|V\|$	$\|\Sigma\|$	d_G	D_G^+	D_G^-	Source
Yeast	7,182	2,361	13	3	60	47	[4]
Cora	91,500	23,166	70	4	104	376	[4]
Wiki	119,882	4,592	120	26	294	1,551	[4]
JDK	150,985	6,434	41	23	375	32,507	[4]
NELL	154,213	75,492	269	2	1,011	1,909	[5]
GP	298,564	144,879	8	2	191	18,553	[3]
Amazon	1,788,725	554,790	82	3	5	549	[6]
ACMCit	9,671,895	1,462,947	72K	7	809	938,039	[1]

TABLE V
PEARSON'S CORRELATION COEFFICIENTS WHEN COMPARING INITIALIZATION FUNCTIONS.

FSim_χ	$\mathsf{FSim_s}$	$\mathsf{FSim_{dp}}$	$\mathsf{FSim_b}$	$\mathsf{FSim_{bj}}$
$\mathcal{L}_I$- $\mathcal{L}_E$	0.990	0.982	0.979	0.969
$\mathcal{L}_I$-$\mathcal{L}_J$	0.967	0.950	0.937	0.922
$\mathcal{L}_J$- $\mathcal{L}_E$	0.985	0.977	0.975	0.962

scores from the graph to itself. $\mathsf{FSim}_\chi\{\theta = a\}$ and $\mathsf{FSim}_\chi\{\mathsf{ub}\}$ denote the computation of FSim_χ uses the optimizations of label-constrained mapping (setting $\theta = a$) and upper-bound updating (Section III-D), respectively. The two optimizations can be meanwhile used as $\mathsf{FSim}_\chi\{\mathsf{ub}, \theta = a\}$. We use $\theta = 0$ by default, which will be omitted for simplicity thereafter.

We implemented FSim_χ in C++. All experiments were conducted on a platform comprising two Intel(R) Xeon(R) CPU E5-2698 v4 @ 2.20GHz (each with 20 cores) and 512GB memory.

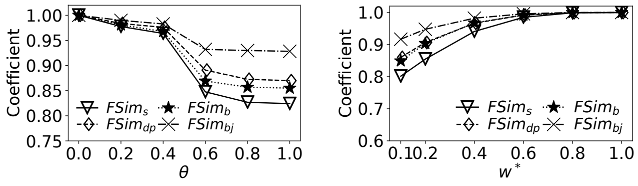
B. Sensitivity Analysis

Our first test was a sensitivity analysis to examine FSim_χ's robustness to parameter tuning and data errors. Following [22], we calculated Pearson's correlation coefficients. The larger the coefficient, the more correlated the evaluated subjects. Note that the patterns were similar across datasets. Hence, only the results for NELL are reported.

Sensitivity of Framework Parameters. We performed the sensitivity analysis against three parameter settings: (1) the initialization function $\mathcal{L}(\cdot)$ presented in Section III-C; (2) the threshold θ for the label-constrained mapping outlined in Remark 2; and (3) the weighting factors outlined in Section III-C.

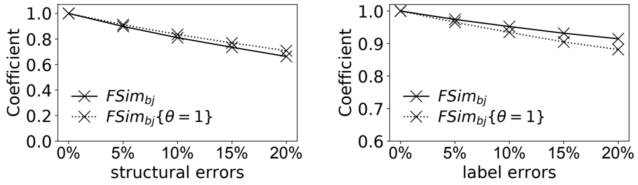
Varying $\mathcal{L}(\cdot)$. In this analysis, we computed and cross-compared the FSim_χ scores using the three different initialization functions: indicator function $\mathcal{L}_I(\cdot)$, normalized edit distance $\mathcal{L}_E(\cdot)$, and Jaro-Winkler similarity $\mathcal{L}_J(\cdot)$. The results are shown in Table V. The Pearson's coefficients for all pairs of initialization functions are very high (> 0.92), which indicates that FSim_χ is not sensitive to initialization functions. Hence, going forward, we used $\mathcal{L}_J(\cdot)$ as the initialization function unless specified otherwise.

Varying θ. For this analysis, we varied θ from 0 to 1 in steps of 0.2, and calculated the Pearson's coefficient against the baseline case of $\theta = 0$ (with w^+ and w^- set to 0.4). The results in Figure 4(a) clearly show that the coefficients decrease as θ increases. This is reasonable as node pairs with $\mathcal{L}(\cdot) < \theta$ will not be considered by the mapping operator. Also, more node pairs are pruned as θ grows. However, the coefficients are still very high (> 0.8) for all variants, even when $\theta = 1$, which indicates that FSim_χ is not sensitive to θ.



(a) varying θ (b) varying w^*

Fig. 4. Pearson's correlation coefficients when varying θ and w^*



(a) varying structural errors (b) varying label errors

Fig. 5. Pearson's correlation coefficients when varying the ratio of data errors

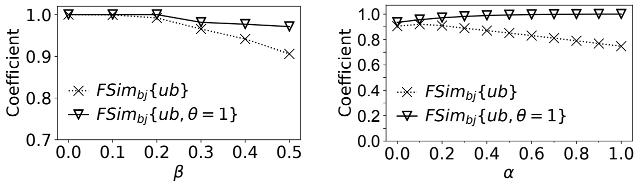
Varying w^*. To examine the influence of the weighting parameters, we varied w^* from 0.1 to 1, where $w^* = 1 - w^+ - w^-$. Recall that $\theta = 1$ constrains mapping only the same-label nodes (Remark 2). As w^* is label-relevant, we computed the coefficients of FSim_χ (vs. $\mathsf{FSim}_\chi\{\theta = 1\}$) by varying w^*. The results, reported in Figure 4, show that the coefficients increase as w^* increases and at $w^* > 0.6$, the coefficient is already almost 1. This is expected because a larger w^* mitigates the impact of the label-constrained mapping. At a more reasonable setting of $w^* = 0.2$, the coefficients sit at around 0.85, which indicates that $\mathsf{FSim}_\chi\{\theta = 1\}$ aligns with FSim_χ well. Hence, we set $w^* = 0.2$ by default in subsequent tests.

Robustness against Data Errors. Figure 5 plots the robustness of $\mathsf{FSim_{bj}}$ against data errors, i.e., structural errors (with edges added/removed) and label errors (with certain labels missing), from one extreme ($\theta = 0$) to the other ($\theta = 1$) as an example of how all simulation variants performed. It is expected that the coefficients decrease as the error level increases. Yet, the coefficients remained high even at the 20% error level (> 0.7 for both cases). This shows that FSim_χ is robust to data errors, which conforms with one of the reasons why we initially thought to propose fractional simulation.

Sensitivity of Upper-bound Updating. To assess the influence of upper-bound updating (Section III-D), we varied α (the approximate ratio) from 0 to 0.5 and β (the threshold) from 0 to 1 in steps of 0.1. Again, the results for all simulation variants were similar, so only the results for $\mathsf{FSim_{bj}}\{\mathsf{ub}\}$ (vs. $\mathsf{FSim_{bj}}$) and$\mathsf{FSim_{bj}}\{\mathsf{ub}, \theta = 1\}$ (vs. $\mathsf{FSim_{bj}}\{\theta = 1\}$) are shown.

Varying β. Figure 6(a) shows the coefficients while varying β from 0 to 0.5 with α fixed to 0.2. It is clear that the coefficients decrease as β increases. This is reasonable as more node pairs are pruned, and the scores become less precise as β gets larger. Note that when $\beta \geq 0.3$, the decreasing trend becomes smoother for $\mathsf{FSim_{bj}}\{\mathsf{ub}, \theta = 1\}$. Observe that even at $\beta = 0.5$, the coefficients are still very high (> 0.9), which indicates that the validity of upper-bound updating is not sensitive to β. We thus set $\beta = 0.5$ going forward to utilize as much pruning power as possible.

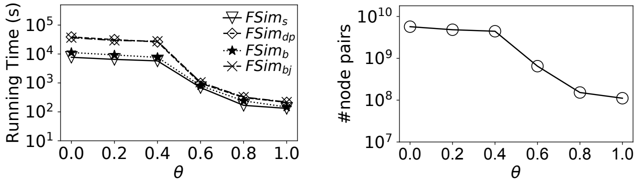
Varying α. Figure 6(b) shows the coefficients when varying α from 0.0 to 1.0. We made two observations here. First, the



(a) varying β

(b) varying α

Fig. 6. Pearson's correlation coefficients when varying α and β



(a) running time

(b) number of node pairs

Fig. 7. Running time of FSim_χ, $\chi \in \{\mathsf{s}, \mathsf{b}, \mathsf{dp}, \mathsf{bj}\}$, while varying θ

coefficients of $\mathsf{FSim}_{\mathsf{bj}}\{\mathsf{ub}\}$ initially increase, then decrease as α gets larger. A possible reason is that $\alpha = 0$ and $\alpha = 1$ are at each extreme of the setting range, but the most appropriate setting lies somewhere in between. Second, the coefficients for $\mathsf{FSim}_{\mathsf{bj}}\{\mathsf{ub}, \theta = 1\}$ increase as α increases. Potentially, the true scores of pruned node pairs are larger than $1 - w^+ - w^-$, and thus a larger α is preferred. Note that when $\alpha = 0$, i.e., when ignoring the pruned node pairs, the coefficients for both $\mathsf{FSim}_{\mathsf{bj}}\{\mathsf{ub}\}$ and $\mathsf{FSim}_{\mathsf{bj}}\{\mathsf{ub}, \theta = 1\}$ were above 0.9; hence, $\alpha = 0$ became our default.

C. *Efficiency*

Varying θ. With NELL as a representative of all tests, Figure 7(a) shows the running time of FSim_χ while varying θ from 0 to 1. The experimental results show that FSim_χ runs faster as θ increases, which is expected since a larger θ contributes to less candidate pairs to compute, as shown in Figure 7(b). We then compared the running time of different simulation variants under certain θ value. It is not surprising that $\mathsf{FSim}_{\mathsf{dp}}$ and $\mathsf{FSim}_{\mathsf{bj}}$ ran slower than the other two variants, as they contain a costly maximum-matching operation (cost analysis in Section IV). $\mathsf{FSim}_{\mathsf{b}}$ ran slower than $\mathsf{FSim}_{\mathsf{s}}$ because the mapping operator of $\mathsf{FSim}_{\mathsf{b}}$ considers both neighbors of a node pair(Table III). At $\theta \geq 0.6$, the difference in running time for all variants was already very small. Considering the sensitivity analysis in Figure 4(a) as well as these results, $\theta = 1$ seems a reasonable setting that renders both good coefficients and performance.

Varying the Datasets. Figure 8 reports the running time of $\mathsf{FSim}_{\mathsf{bj}}$, the most costly simulation variant, with different optimizations on all datasets. Additionally, experiments that resulted in out-of-memory errors have been omitted. From these tests, we made the observations: (1) the upper-bound updating alone contributed about 5× the performance gain compared to $\mathsf{FSim}_{\mathsf{bj}}\{\mathsf{ub}\}$ with $\mathsf{FSim}_{\mathsf{bj}}$. (2) Label-constrained mapping is the most effective optimization, making $\mathsf{FSim}_{\mathsf{bj}}\{\theta = 1\}$ faster than $\mathsf{FSim}_{\mathsf{bj}}$ by up to 3 orders of magnitude. Applying both label-constrained mapping and upper-bound updating, $\mathsf{FSim}_{\mathsf{bj}}\{\mathsf{ub}, \theta = 1\}$ was the only algorithm that could complete the executions on all datasets in time, including the two largest ones, Amazon and ACMCit.

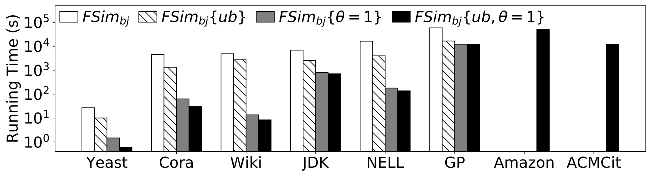
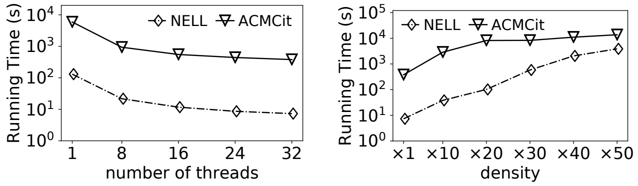


Fig. 8. Running time of $\mathsf{FSim}_{\mathsf{bj}}$ on all datasets with different optimizations



(a) varying the number of threads

(b) varying density

Fig. 9. Parallelization and Scalability

Parallelization and Scalability. We studied the scalability of FSim_χ with parallelization on two representative datasets, i.e., NELL and ACMCit (with more than 1 million nodes). The results for $\mathsf{FSim}_{\mathsf{bj}}\{\mathsf{ub}, \theta = 1\}$ follow.

Varying the Number of Threads. Figure 9(a) shows the running time of $\mathsf{FSim}_{\mathsf{bj}}\{\mathsf{ub}, \theta = 1\}$ by varying the number of threads from 1 to 32. We observe that both curves demonstrate reasonable decreasing trends as the number of threads increases. The benefits from 1 to 8 threads are substantial. After 8, the reward ratio flattens due to the cost of thread scheduling. Specifically, when setting $t = 32$, parallelization can speed up the computation by 15 to 17 times of magnitude.

Varying Density. Figure 9(b) reports the running time of $\mathsf{FSim}_{\mathsf{bj}}\{\mathsf{ub}, \theta = 1\}$ (with 32 threads) while varying the density of the datasets from ×10 to ×50 by randomly adding edges. Unsurprisingly, the running times of both grew longer as the graphs became denser. However, although increased density means greater computational complexity in theory, it also means each node has more neighbors by expectation. Hence, the upper bound in Equation 6 may become smaller, which, in turn, contributes to greater upper-bound pruning power. This may offset some of the increase in computation complexity. Note that FSim_χ finished within reasonable time on the ACMCit with 50× more edges, indicating that it is scalable to the graphs with hundreds of millions of edges.

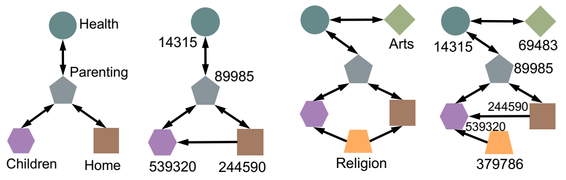
D. *Case Studies*

In this subsection, we used three case studies to exhibit the potential of FSim_χ in the applications of pattern matching, node similarity measurement and RDF graph alignment. We will demonstrate the following strengths of our framework.

S1. Our FSim_χ framework quantifies the degree of simulation, which remedies the coarse "yes-or-no" semantics of simulation, significantly improves the effectiveness, and expand the scope of applying simulation.

S2. When multiple simulation variants are suitable for a certain application, the FSim_χ framework provides a flexible way to experiment with all suitable variants, so as to determine the one that performs the best.

Before simply driving into the case studies, the first question to answer is: which simulation variant should be used for



(a) Q_1 (b) G_1 (c) Q_2 (d) G_2

Fig. 10. Real-life matches on the Amazon graph. G_1 and G_2 are top-1 matches of FSim$_\chi$ for answering queries Q_1 and Q_2 respectively. Nodes in match G are marked by their item ids, while nodes in query Q are marked by their labels. Nodes with the same shape have the same label.

a given application? We discuss the answer intuitively. Subgraph pattern matching is essentially asymmetric (matching the pattern graph to the data graph but not the other way around), and thus FSim$_{\mathsf{s}}$ and FSim$_{\mathsf{dp}}$ are appropriate choices. Node similarity measurement and graph alignment require symmetry, and hence FSim$_{\mathsf{b}}$ and FSim$_{\mathsf{bj}}$ are applied. The codes of all the baselines were provided by the respective authors. $\mathcal{L}(\cdot)$ was used indicator function since the semantics of node labels in the studied data were clear and without ambiguity.

Pattern Matching. In this case study, we first considered strong simulation (exact simulation by nature, [30]) and dp-simulation [40] as two baselines, and compared them with FSim$_{\mathsf{s}}$ and FSim$_{\mathsf{dp}}$ to illustrate how FSim$_\chi$ facilitates pattern matching. Figure 10 shows two example matches on the Amazon graph (see Table IV for graph statistics). When answering query Q_1, strong simulation (and dp-simulation) returns G_1, pictured in Figure 10(b), which is also the top-1 result of FSim$_{\mathsf{s}}$ (and FSim$_{\mathsf{dp}}$). Clearly, a simulation relation exists between Q_1 and G_1, and FSim$_\chi$ captures G_1 with the highest score because of simulation definiteness (Definition 4). Q_2 adds two extra nodes with new labels to Q_1 but, with this modification, both strong simulation and dp-simulation fail to return a result while FSim$_\chi$ returns G_2 (strength S1), which closely matches Q_2 by missing only an edge.

For a more complete study, we also compared the results of FSim$_\chi$ with some other approximate pattern matching algorithms. The related algorithms can be summarized in two categories: (1) the edit-distance based algorithms, e.g., SAPPER [49] and TSpan [51], which enumerate all matches with mismatched edges up to a given threshold; and (2) the similarity-based algorithms that compute matches based on (sub)graph similarity or node similarity. To name a few, G-Ray [43] computes the "goodness" of a match based on node proximity. IsoRank [39], NeMa [25] and NAGA [14] find matches based on node similarity. G-Finder [27] and SAGA [42] design cost functions with multiple components to allow node mismatches and graph structural differences. SLQ [46] and S^4 [50] find matches in RDF knowledge graphs by considering the semantics of queries. More specifically, S^4 uses the semantic graph edit distance to integrate structure similarity and semantic similarity. Note that, in the Amazon graph, an edge from u to v indicates that people are highly likely to buy item v after buying item u [30], and hence there is no complex semantics among edges. As a result, we choose TSpan, NAGA and G-Finder, the state-of-the-art algorithms in each category, as another three baselines.

We followed the state-of-the-art algorithm NAGA [14] for match generation and quality evaluation. Briefly, node pairs with high FSim$_\chi$ scores are considered to be "seeds", and matches are generated by expanding the regions around the "seeds" subsequently. The evaluated queries are generated randomly by extracting subgraphs from the data graph and introducing structural noises (randomly insert edges, up to 33%) or label noises (randomly modify node labels, up to 33%). We then evaluated different algorithms across four query scenarios: (1) queries with no noises (Exact); (2) queries with structural noises only (Noisy-E); (3) queries with label noises only (Noisy-L); and (4) queries with both kinds of noises (Combined). Note that the queries are extracted from the graphs, which naturally serve as the "ground truth". Given a query Q and a returned match ϕ (we use top-1 match in this case study), the F_1 score is calculated by $F_1 = \frac{2 \cdot P \cdot R}{(P+R)}$, where $P = \frac{|\phi_t|}{|\phi|}$, $R = \frac{|\phi_t|}{|Q|}$, ϕ_t is a subset of ϕ that includes the correctly discovered node matches in ϕ, and $|X|$ indicates the number of nodes in the match or graph, $\forall X \in \{\phi_t, \phi, Q\}$.

TABLE VI
AVERAGE F1 SCORES (%) WHILE ANSWERING QUERIES IN DIFFERENT SCENARIOS ON THE AMAZON DATASET. TSpan-x INDICATES MISS-MATCHING UP TO x EDGES IN TSpan.

Query Scenario	Baselines					FSim$_\chi$	
	NAGA	G-Finder	TSpan-1	TSpan-3	Strong Simulation	FSim$_{\mathsf{s}}$	FSim$_{\mathsf{dp}}$
Exact	30.2	**100**	**100**	**100**	**100**	**100**	**100**
Noisy-E	30.5	49.2	71.0	**95.8**	50.0	84.0	65.7
Noisy-L	20.6	40.7	-	-	33.3	**75.1**	73.2
Combined	21.2	40.9	-	-	29.2	**76.6**	66.7

Table VI[4] shows the F1 scores of different algorithms. The result is an average from 100 random queries of sizes ranging from 3 to 13. dp-simulation was not compared as it is similar to strong simulation. As with the last results, strong simulation performed poorly against noise. In comparison, FSim$_\chi$ was more robust and performed much better (strength S1). Additionally, FSim$_{\mathsf{s}}$ outperformed NAGA, G-Finder and TSpan-1 by a big margin on all query scenarios. TSpan-3 performed well in "Exact" and "Noisy-E" with its highest F1 score of 95.8% for "Noisy-E". This is because TSpan-3 finds all matches with up to 3 mismatched edges, which is not less than the number of noisy edges in most queries. However, TSpan favors the case with missing edges rather than nodes. Thus, it has no results for "Noisy-L" and "Combined". In summary, FSim$_\chi$ is qualified for approximate pattern matching (strength S1). While both s- and dp-simulation can be configured for the application, FSim$_{\mathsf{s}}$ is more robust to noises and performs better than FSim$_{\mathsf{dp}}$ (strength S2).

Node Similarity Measurement. In this case study, we compared FSim$_\chi$ to four state-of-the-art similarity measurement algorithms: PCRW [26], PathSim [41], JoinSim [45] and nSimGram [12]. Following [12], [41], we used the DBIS dataset, which contains 60,694 authors, 72,902 papers and 464 venues. In DBIS, the venues and papers are labeled as "V" and "P", respectively. The authors are labeled by their names.

We first computed the top-5 most similar venues to WWW using all algorithms. The results are shown in Table VII. Note that WWW$_1$, WWW$_2$ and WWW$_3$ all represent the WWW venue but with different node ids in DBIS, and thus they

[4] The results of NAGA are provided by the authors and we acknowledge the assistance from Dr. Sourav Dutta and Dr. Shubhangi Agarwal.

TABLE VII
THE TOP-5 SIMILAR VENUES FOR "WWW" OF DIFFERENT ALGORITHMS

Rank	PCRW	PathSim	JoinSim	nSimGram	$\mathsf{FSim_b}$	$\mathsf{FSim_{bj}}$
1	WWW	WWW	WWW	WWW	WWW	WWW
2	SIGIR	CIKM	WWW_1	CIKM	CIKM	WWW_1
3	ICDE	SIGKDD	CIKM	SIGIR	ICDE	CIKM
4	VLDB	WISE	WSDM	WWW_1	VLDB	WWW_2
5	Hypertext	ICDM	WWW_2	SIGKDD	SIGIR	WWW_3

TABLE VIII
NDCG RESULTS OF NODE SIMILARITY ALGORITHMS

Baselines				Fractional χ-simulation	
PCRW	PathSim	JoinSim	nSimGram	$\mathsf{FSim_b}$	$\mathsf{FSim_{bj}}$
0.684	0.684	0.689	0.700	0.699	**0.733**

are naturally similar to WWW. Although all algorithms gave reasonable results, $\mathsf{FSim_{bj}}$ was the only one to return WWW_1, WWW_2 and WWW_3 in the top-5 results. In addition, if we applied exact b- and bj-simulation to the task, other than "WWW" itself ("Yes"), all the other venues had the same score ("No"). This shows that FSim_χ can be applied to the scenarios that require fine-grained evaluation, such as node similarity measurement (strength S1).

Following [12], [41], we further computed the top-15 most similar venues to 15 subject venues (same as [12]) of each algorithm. For each subject venue, we labeled each returned venue with a relevance score: 0 for non-relevant, 1 for some-relevant, and 2 for very-relevant, considering both the research area and venue ranking in [2]. For example, the relevance score for ICDE and VLDB is 2 as both are top-tier conferences in the area of database. We then evaluated the ranking quality of the algorithms using nDCG (the larger the score, the better).

Table VIII shows the nDCG results. Accordingly, FSim_χ outperforms the state-of-the-art algorithms by a large margin. This indicates that FSim_χ is qualified to measure node similarity on labeled graphs (strength S1). The result that $\mathsf{FSim_{bj}}$ outperforms $\mathsf{FSim_b}$ in both the "WWW" case and the general evaluation suggests $\mathsf{FSim_{bj}}$ is a better candidate for similarity measurement (strength S2).

RDF Graph Alignment. We investigate the potential of FSim_χ in RDF graph alignment and briefly discuss its performance below. We followed Olap [11] (a bisimulation-based alignment algorithm) to align three different versions of biological graphs from different times, G_1, G_2 and G_3 [3]. G_1 has 133,195 nodes and 273,512 edges, G_2 has 138,651 nodes and 285,000 edges, G_3 includes 144,879 nodes and 298,564 edges, and all of them have 8 node labels and 23 edge labels. Note that the original URI values in these datasets do not change over time. Hence, we can use this information to identify the ground truth alignment. In addition to Olap, we also included another four state-of-the-art algorithms, namely k-bisimulation [44], GSANA [47], FINAL [48] and EWS [24]. When aligning graphs with FSim_χ, a node $u \in V_1$ will be aligned to a node set $A_u = \text{argmax}_{v \in V_2} \mathsf{FSim}_\chi(u, v)$, while with k-bisimulation, u will be aligned to $A_u = \{v | v \in V_2 \wedge u \text{ and } v \text{ are bisimilar}\}$. The F1 score of FSim_χ and k-bisimulation is calculated by $F1 = \sum_{u \in V_1} \frac{2 \cdot P_u \cdot R_u}{|V_1|(P_u + R_u)}$, where P_u (resp. R_u) is $\frac{1}{|A_u|}$ (resp. 1) if A_u contains the ground truth, and 0 otherwise. We follow the settings in the related papers for the other baselines.

Table IX reports the F1 scores of each algorithm. Note that we also tested the bisimulation, which resulted in 0% F1 scores in both cases since there is no exact bisimulation relation between two graphs. k-bisimulation performs better than bisimulation as it, to some extent, approximates bisimulation. From Table IX, our FSim_χ had the highest F1 scores and thus outperformed all the other baselines. This shows that we can apply $\mathsf{FSim_b}$ and $\mathsf{FSim_{bj}}$ with high potential for graph alignment (strength S1). $\mathsf{FSim_b}$ outperforms $\mathsf{FSim_{bj}}$ and thus is a better candidate for graph alignment (strength S2).

TABLE IX
THE F1 SCORES (%) OF EACH ALGORITHM WHEN ALIGNING TWO GRAPHS. x-BISIM INDICATES SETTING $k = x$ IN k-BISIMULATION.

Graphs	Baselines						FSim_χ	
	2-bisim	4-bisim	Olap	GSANA	FINAL	EWS	$\mathsf{FSim_b}$	$\mathsf{FSim_{bj}}$
G_1-G_2	19.9	9.1	37.9	11.8	55.2	70.8	**97.6**	96.5
G_1-G_3	53.0	10.9	37.6	14.9	52.7	65.3	**96.9**	95.6

Efficiency Evaluation. Given the superior effectiveness of FSim_χ in the above case studies, one may also be interested in its efficiency. Next, we show the running time of FSim_χ (with 32 threads) and the most effective baseline in each case study. We will also report the running time of exact simulation (or its variant) if it is applied and effective in the case study. For pattern matching, FSim_χ on average took 0.25s for each query. In comparison, exact simulation took around 1.2s, and TSpan, the most effective baseline, spent more than 70s. In similarity measurement, nSimGram took 0.03ms to compute a single node pair, while FSim_χ finished the computation within 6500s for 134060×134060 pairs or roughly 0.0004ms per pair. In graph alignment, k-bisimulation ($k = 4$) spent 0.4s for the computation, and EWS spent 1496s. Our FSim_χ ran a bit slower than EWS and took 3120s, which is tolerable as it is much more effective than the other algorithms. Note that it is not straightforward and potentially unfair to compare with all the baselines as they either focus on per-query computation (e.g., PathSim and JoinSim) or have been individually implemented in different languages (e.g., Olap in Python and FINAL in Matlab).

VI. RELATED WORK

Simulation and Its Variants. In this paper, we focused on four simulation variants: simple simulation [32], [30], bisimulation [33], degree-preserving simulation [40] and bijective simulation. The original definition of simulation [32] only considered out-neighbors, but Ma et al.'s redefinition in 2011 [30] takes in-neighbors into account and hence is the definition we used. Reverting to the original definition is as easy as setting $w^- = 0$ in our framework. Additionally, we discussed a variant of approximate bisimulation, namely k-bisimulation [8], [28], [29], [44], and investigated its relation to our framework (Section IV-C). There are other variants that have not yet included in the framework, including bounded simulation [15] and weak simulation [33]. These variants consider the k-hop neighbors ($k \geq 1$) in addition to the immediate neighbors. As an interesting future work, we will study to incorporate them in our framework. There are also some algorithms that aim to compute simulation (variants) efficiently and effectively, e.g., a hash-based algorithm in [44], external-memory algorithms in [19], [29], a distributed algorithm in [28] and a partition refinement algorithm in [35]. However, all these algorithms compute the "yes-or-no" simulation (or its variant) and cannot provide fractional scores as proposed in this paper.

Node Similarity Measures. We have shown that $\mathsf{FSim_{bj}}$ is qualified for node similarity measurement. Thus, we review node similarity measures on labeled graphs. SimRank [21] and RoleSim [22] are two representative measures, and their relations to our FSim_χ have been discussed in Section IV-C. As these two measures are less effective in computing node similarity on labeled graphs [12], [41], similarity measures [12], [20], [41], [45] were proposed. PathSim [41], for instance, uses a ratio of meta-paths connecting two nodes as the measure. JoinSim [45] is similar to PathSim, but it satisfies the triangle inequality. nSimGram [12] computes node similarity based on q-grams instead of meta-paths to capture more topology information. Note that these measures cannot substitute our work as their scores are not related to simulation and thus are not suitable to quantify the extent of simulation.

Similarity-based Applications. There are a number of works on pattern matching and graph alignment that are based on node similarity techniques. These works may differ in measuring node similarities. Specifically, IsoRank [39] computes the similarity between two nodes based on an weighted average of their neighbors' scores. NeMa [25] defines a vector that encodes the neighborhood information for each node. The distance between two nodes is then computed from these vectors. NAGA [14] leverages statistical significance through chi-square measure to compute node similarity. REGAL [18] measures the similarity of two nodes by taking the information of k-hop neighbors into account. FIRST [13] and FINAL [48] use a Sylvester equation to compute similarities, which encodes structural consistency and attribute consistency of two networks. For similar reasons, these works are also not suitable to quantify the degree of simulation.

VII. Conclusion

In this paper, we formally define fractional χ-simulation to quantify the degree to which one node simulates another by a χ-simulation. We then propose the FSim_χ computation framework to realize the quantification for all χ-simulations. We conduct extensive experiments to demonstrate the effectiveness and efficiency of the fractional χ-simulation framework. Considering end-users are also interested in the top-k similarity search. In the future, we plan to devise efficient techniques to process top-k queries based on the FSim_χ.

References

[1] ArnetMiner: https://www.aminer.org/data.

[2] Core rankings portal: http://portal.core.edu.au/conf-ranks/.

[3] The guide to pharmacology: https://www.guidetopharmacology.org.

[4] KONECT: http://konect.uni-koblenz.de/.

[5] NELL: https://github.com/xwhan/DeepPath.

[6] SNAP: http://snap.stanford.edu/data.

[7] Solo: https://en.wikipedia.org/wiki/Solo:_A_Star_Wars_Story.

[8] L. Aceto, A. Ingólfsdóttir, and J. Srba. The algorithmics of bisimilarity. In *Advanced Topics in Bisimulation and Coinduction*. 2012.

[9] D. Avis. A survey of heuristics for the weighted matching problem. *Networks*, 13(4):475–493, 1983.

[10] L. Babai. Graph isomorphism in quasipolynomial time. In *ACM Symposium on Theory of Computing*, pages 684–697, 2016.

[11] P. Buneman and S. Staworko. RDF graph alignment with bisimulation. *PVLDB*, 9(12):1149–1160, 2016.

[12] A. Conte, G. Ferraro, R. Grossi, A. Marino, K. Sadakane, and T. Uno. Node similarity with q-grams for real-world labeled networks. In *SIGKDD*, pages 1282–1291, 2018.

[13] B. Du, S. Zhang, N. Cao, and H. Tong. FIRST: fast interactive attributed subgraph matching. In *SIGKDD*, pages 1447–1456, 2017.

[14] S. Dutta, P. Nayek, and A. Bhattacharya. Neighbor-aware search for approximate labeled graph matching using the chi-square statistics. In *WWW*, pages 1281–1290, 2017.

[15] W. Fan, J. Li, S. Ma, N. Tang, Y. Wu, and Y. Wu. Graph pattern matching: From intractable to polynomial time. *PVLDB*, 3(1):264–275, 2010.

[16] W. Fan, J. Li, X. Wang, and Y. Wu. Query preserving graph compression. In *SIGMOD*, pages 157–168, 2012.

[17] G. H. L. Fletcher, D. V. Gucht, Y. Wu, M. Gyssens, S. Brenes, and J. Paredaens. A methodology for coupling fragments of xpath with structural indexes for XML documents. *Inf. Syst.*, 34(7):657–670, 2009.

[18] M. Heimann, H. Shen, T. Safavi, and D. Koutra. REGAL: representation learning-based graph alignment. In *ACM CIKM*, pages 117–126, 2018.

[19] J. Hellings, G. H. L. Fletcher, and H. J. Haverkort. Efficient external-memory bisimulation on dags. In *SIGMOD*, pages 553–564, 2012.

[20] Z. Huang, Y. Zheng, R. Cheng, Y. Sun, N. Mamoulis, and X. Li. Meta structure: Computing relevance in large heterogeneous information networks. In *SIGKDD*, pages 1595–1604, 2016.

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

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

[23] R. Kaushik, P. Bohannon, J. F. Naughton, and H. F. Korth. Covering indexes for branching path queries. In *SIGMOD*, pages 133–144, 2002.

[24] E. Kazemi, S. H. Hassani, and M. Grossglauser. Growing a graph matching from a handful of seeds. *PVLDB*, 8(10):1010–1021, 2015.

[25] A. Khan, Y. Wu, C. C. Aggarwal, and X. Yan. Nema: Fast graph search with label similarity. *PVLDB*, 6(3):181–192, 2013.

[26] N. Lao and W. W. Cohen. Relational retrieval using a combination of path-constrained random walks. *Machine Learning*, 81(1):53–67, 2010.

[27] L. Liu, B. Du, J. Xu, and H. Tong. G-finder: Approximate attributed subgraph matching. In *IEEE Big Data*, pages 513–522, 2019.

[28] Y. Luo, Y. de Lange, G. H. L. Fletcher, P. D. Bra, J. Hidders, and Y. Wu. Bisimulation reduction of big graphs on mapreduce. In *BNCOD*, 2013.

[29] Y. Luo, G. H. L. Fletcher, J. Hidders, Y. Wu, and P. D. Bra. External memory k-bisimulation reduction of big graphs. In *CIKM*, 2013.

[30] S. Ma, Y. Cao, W. Fan, J. Huai, and T. Wo. Capturing topology in graph pattern matching. *PVLDB*, 5(4):310–321, 2011.

[31] S. Ma, Y. Cao, W. Fan, J. Huai, and T. Wo. Strong simulation: Capturing topology in graph pattern matching. *TODS*, 39(1):4:1–4:46, 2014.

[32] R. Milner. An algebraic definition of simulation between programs. In *IJCAI*, pages 481–489, 1971.

[33] R. Milner. *Communication and concurrency*. 1989.

[34] P. Ramanan. Covering indexes for XML queries: Bisimulation - simulation = negation. In *VLDB*, pages 165–176, 2003.

[35] F. Ranzato. An efficient simulation algorithm on kripke structures. *Acta Informatica*, 51(2):107–125, 2014.

[36] Y. Sasaki, G. Fletcher, and M. Onizuka. Structural indexing for conjunctive path queries. *CoRR*, 2020.

[37] A. Schätzle, A. Neu, G. Lausen, and M. Przyjaciel-Zablocki. Large-scale bisimulation of RDF graphs. In *SWIM@SIGMOD*, 2013.

[38] N. Shervashidze, P. Schweitzer, E. J. van Leeuwen, K. Mehlhorn, and K. M. Borgwardt. Weisfeiler-lehman graph kernels. *JMLR*, 2011.

[39] R. Singh, J. Xu, and B. Berger. Global alignment of multiple protein interaction networks with application to functional orthology detection. *Proc. Natl. Acad. Sci.*, 105(35):12763–12768, 2008.

[40] C. Song, T. Ge, C. X. Chen, and J. Wang. Event pattern matching over graph streams. *PVLDB*, 8(4):413–424, 2014.

[41] Y. Sun, J. Han, X. Yan, P. S. Yu, and T. Wu. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *PVLDB*, 4(11):992–1003, 2011.

[42] Y. Tian, R. C. McEachin, C. Santos, D. J. States, and J. M. Patel. SAGA: a subgraph matching tool for biological graphs. *Bioinform.*, 2007.

[43] H. Tong, C. Faloutsos, B. Gallagher, and T. Eliassi-Rad. Fast best-effort pattern matching in large attributed graphs. In *SIGKDD*, 2007.

[44] W. van Heeswijk, G. H. L. Fletcher, and M. Pechenizkiy. On structure preserving sampling and approximate partitioning of graphs. In *ACM Symposium on Applied Computing*, pages 875–882, 2016.

[45] Y. Xiong, Y. Zhu, and P. S. Yu. Top-k similarity join in heterogeneous information networks. *TKDE*, 27(6):1710–1723, 2015.

[46] S. Yang, Y. Wu, H. Sun, and X. Yan. Schemaless and structureless graph querying. *PVLDB*, 7(7):565–576, 2014.

[47] A. Yasar and Ü. V. Çatalyürek. An iterative global structure-assisted labeled network aligner. In *SIGKDD*, pages 2614–2623, 2018.

[48] S. Zhang and H. Tong. FINAL: fast attributed network alignment. In *SIGKDD*, pages 1345–1354, 2016.

[49] S. Zhang, J. Yang, and W. Jin. SAPPER: subgraph indexing and approximate matching in large graphs. *PVLDB*, 3(1):1185–1194, 2010.

[50] W. Zheng, L. Zou, W. Peng, X. Yan, S. Song, and D. Zhao. Semantic SPARQL similarity search over RDF knowledge graphs. *PVLDB*, 2016.

[51] G. Zhu, X. Lin, K. Zhu, W. Zhang, and J. X. Yu. Treespan: efficiently computing similarity all-matching. In *SIGMOD*, pages 529–540, 2012.