Graph Pattern Matching Preserving Label-Repetition Constraints

Houari Mahfoud

1 Abou-Bekr Belkaid University & LRIT Laboratory, Tlemcen, Algeria
houari.mahfoud@gmail.com

Abstract

Graph pattern matching is a routine process for a wide variety of applications such as social network analysis. It is typically defined in terms of subgraph isomorphism which is NP-Complete. To lower its complexity, many extensions of graph simulation have been proposed which focus on some topological constraints of pattern graphs that can be preserved in polynomial-time over data graphs. We discuss in this paper the satisfaction of a new topological constraint, called Label-Repetition constraint. To the best of our knowledge, existing polynomial approaches fail to preserve this constraint, and moreover, one can adopt only subgraph isomorphism for this end which is cost-prohibitive. We present first a necessary and sufficient condition that a data subgraph must satisfy to preserve the Label-Repetition constraints of the pattern graph. Furthermore, we define matching based on a notion of triple simulation, an extension of graph simulation by considering the new topological constraint. We show that with this extension, graph pattern matching can be performed in polynomial-time, by providing such an algorithm. Our algorithm is sub-quadratic in the size of data graphs only, and quartic in general. We show that our results can be combined with orthogonal approaches for more expressive graph pattern matching.

1998 ACM Subject Classification F.2 [Analysis of algorithms and problem complexity]: Non-numerical algorithms and problems[Pattern matching]

Keywords and phrases Graph pattern matching, triple simulation, Label-Repetition constraint

Introduction

Modeling data with graphs is one of the most active topics in the database community these days. This model has recently gained wide applicability in numerous domains that find the relational model too restrictive, such as social networks [5], biological networks, Semantic Web, crime detection networks and many others. Indeed, it is less complex and also most natural for users to reason about an increasing number of popular datasets, such as the underlying networks of Twitter, Facebook, or LinkedIn, within a graph paradigm. In emerging applications such as social networks, edges of data graphs (resp. pattern graphs) can be typed [6] to denote various relationships such as marriage, friendship, recommendation, co-membership, etc. Moreover, pattern graphs can define multi-labeled vertices [18] to look, e.g., for persons with different possible profiles.

Given a data graph G and a pattern graph Q, the problem of graph pattern matching is to find all subgraphs of G that satisfy both the labeling properties and topological constraints carried by Q. Matching here is expressed in terms of subgraph isomorphism which consists to find all subgraphs of G that are isomorphic to Q. Graph pattern matching via subgraph isomorphism is an NP-Complete problem as there are possibly an exponential number of subgraphs in G that match Q. To tackle this NP-Completeness, graph simulation [17] has
been adopted for graph pattern matching \cite{16} to preserve child-relationships only. Unlike subgraph isomorphism which requires a \textit{bijective} mapping function from pattern nodes to data nodes, graph simulation is defined by a simple \textit{binary} relation which can be computed in quadratic time. A cubic-time extension of graph simulation, called \textit{strong simulation}, has been proposed \cite{14} by enforcing two additional conditions: \textit{duality} to preserve child and parent relationships of the pattern graph; and \textit{locality} to overcome excessive matching by considering only subgraphs that have radius bounded by the diameter of the pattern graph.

Nonetheless, the polynomial-time complexity comes at a price: the result of strong simulation may contain incorrect matches as shown below.

\textbf{Example 1.} Consider the real-life example taken from \cite{14} with minor modification. A headhunter (HR) wants to find a biologist (BIO) to help a group of software engineers (SE) analyze genetic data. To do this, she uses the network $G$ depicted in Fig. 1. In $G$, nodes denote persons with different profiles, and edges indicate recommendations between these persons. The cycle between the nodes $d_9$ and $d_{12}$ contains many DM (data mining specialist) that are all connected to the BIO represented by the node $d_2$. The biologist BIO to find is specified with the pattern graph $Q_1$ of Fig. 1. Intuitively, the BIO has to be recommended by: (a) an HR person since the headhunter trusts the judgment of a person with the same occupation; (b) at least two SE that are recommended by the same HR person (to increase incredibility), that is, the BIO has a strong experience by working with different SEs; and (c) a DM, as data mining techniques are required for the job. Moreover, there is an artificial intelligence expert (AI) who recommends the DM and is recommended by a DM.

When strong simulation is adopted, the subgraph $G_2$ of $G$ is returned as the only match of $Q_1$ in $G$. However, the BIO of this match, represented by the node $d_1$, is recommended by only one SE, which is incorrect w.r.t $Q_1$. To make search less restrictive, one can look for a BIO with the same constraints specified by $Q_1$ excepting that this BIO can be recommended by only one SE. This search is specified by the pattern graph $Q_2$ of the same figure. In this case, strong simulation returns $G_2$ as the only match of $Q_2$ in $G$, which is a correct. Notice however that strong simulation does not make difference between $Q_1$ and $Q_2$ since the two pattern graphs are matched over $G$ to the same match result.

The pattern graph $Q_1$ illustrates a new kind of topology that we call \textit{Label-Repetition (LR)} constraint. Graph simulation \cite{16} and its counterparts \cite{7,14} fail to preserve this constraint. One can adopt subgraph isomorphism to preserve LR constraints during graph pattern matching. The challenge is that subgraph isomorphism is NP-Complete and real-life data graphs are often big, e.g., the social graph of Facebook has billions of nodes and trillions of

\begin{figure}
\centering
\includegraphics[width=\textwidth]{figure1.pdf}
\caption{Querying a recommendation network.}
\end{figure}
edges \[11\]. This motivates us to study an extension of graph simulation in order to preserve \( LR \) constraints in polynomial-time.

**Contributions & Road-map.** Our main contributions are as follows:

1. We introduce a new extension of graph simulation, called **triple simulation**, to preserve \( LR \) constraints (Section 3).
2. We define a necessary and sufficient condition that characterizes the satisfaction of \( LR \) constraints and we compute its time complexity (Section 4).
3. We develop a graph pattern matching algorithm which requires a polynomial-time to preserve \( Child \) and \( Parent \) relationships, as well as \( LR \) constraints (Section 5). Finally, we show how to improve the quality of our match results by using the notion of locality (Section 6).

**Related work.** We categorize related work as follows.

- **Polynomial-time graph pattern matching:** Traditional matching is by subgraph isomorphism, which is NP-Complete \[3\] and found often too restrictive to capture sensible matches \[2\]. To loosen the restriction, one direction is to adopt graph simulation \[17\]. Matching based on graph simulation \[16\] preserves only child relationships of the pattern graphs, which makes it useful for some applications like Web sites classification \[1\]. In other applications however, e.g. social network analysis, the result of such matching may have a structure drastically different from that of the pattern graph, and often very large to analysis and understand.

  To handle this, **strong simulation** is proposed \[14\] to capture child and parent relationships (notion of **duality**), and to make match results bounded by the diameter of the underlying pattern graph (notion of **locality**). This approach has proven efficient since it is in PTIME. However, it can not match correctly pattern graphs with \( LR \) constraints.

- **Quantified pattern graphs:** Closer to our work is \[10\] that introduces quantified pattern graphs (QGPs), an extension of pattern graphs by supporting simple counting quantifiers on edges. A QGP naturally expresses numeric and ratio aggregates, and negation besides existential and universal quantification. Notice that any ratio aggregate can be translated into numeric aggregate. Quantified matching is NP-Complete in the absence of negation and DP-Complete for general QGPs. As shown in the Appendix \[17\] any QGP with numeric aggregates can be translated into a simple pattern graph with only \( LR \) constraints. This translation allows to preserve numeric and ratio aggregates on edges in polynomial-time, contrary to the prohibitive-cost found by the authors \[10\]. Furthermore, we think that matching over pattern graphs with negation and universal quantifications on edges can be done in PTIME if treated as an extension of graph simulation (one of our future directions).

## 2 Background

We give basic notions of graphs and then we review some graph pattern matching approaches.

**Graphs.** A **directed graph** (or simply a graph) is defined with \( G(V, E, \lambda) \) where: 1) \( V \) is a finite set of nodes; 2) \( E \subseteq V \times V \) is a finite set of edges in which \((u, u')\) denotes an edge from nodes \( u \) to \( u' \); and 3) \( \lambda \) is a labeling function that maps each node \( u \in V \) to a label \( \lambda(u) \) in a set \( \sum(G) \) of labels. We simply denote \( G \) as \( (V, E) \) when it is clear from the context.

In this paper, both data graphs and pattern graphs are specified with the previous graph structure. Moreover, we assume that pattern graphs are connected, as a common practice.

**Distance and diameter** \[14\]. The **distance** from nodes \( n \) to \( n' \) in a graph \( G \), denoted by \( \text{dist}(n, n') \), is the length of the shortest undirected path from \( n \) to \( n' \) in \( G \). The **diameter**

\footnote{The proofs are given in Appendix.}
of a connected graph $G$, denoted by $d_G$, is the longest shortest distance of all pairs of nodes in $G$, that is, $d_G = \max(\text{dist}(n, n'))$ for all nodes $n, n'$ in $G$.

**Graph pattern matching.** A data graph $G(V, E, \lambda)$ may match a pattern graph $Q(V_Q, E_Q, \lambda_Q)$ via different methods.

**A)** *Subgraph isomorphism:* A subgraph $G_s(V_s, E_s, \lambda_s)$ of $G$ matches $Q$ via subgraph isomorphism, denoted $G_s \sim_{sio} Q$, if there exists a bijective function $f: V_Q \rightarrow V_s$ such that: 1) for each node $n \in V_Q$, $\lambda_s(n) = \lambda_s(f(n))$; and 2) for each edge $(n, n') \in E_Q$, there exists an edge $(f(n), f(n')) \in E_s$.

**B)** *Graph simulation:* $G$ matches $Q$ via graph simulation [16], denoted $Q \prec G$, if there exists a binary match relation $S \subseteq V_Q \times V$ such that:
1. For each $(u, v) \in S$, $\lambda_Q(u) = \lambda(v)$; and
2. For each node $u \in V_Q$, there exists a node $v \in V$ such that: a) $(u, v) \in S$; and b) for each edge $(u, u') \in E_Q$, there exists an edge $(v, v') \in E$ with $(u', v') \in S$.

Intuitively, graph simulation preserves only child relationships of the pattern graph.

**C)** *Dual simulation:* $G$ matches $Q$ via dual simulation [13], denoted $Q \prec_D G$, if there exists a binary match relation $S_D \subseteq V_Q \times V$ such that:
1. For each $(u, v) \in S_D$, $\lambda_Q(u) = \lambda(v)$; and
2. For each node $u \in V_Q$, there exists a node $v \in V$ such that: a) $(u, v) \in S_D$; b) for each edge $(u, u') \in E_Q$, there exists an edge $(v, v') \in E$ with $(u', v') \in S_D$; and moreover c) for each edge $(u, u) \in E_Q$, there exists an edge $(v, v') \in E$ with $(v', v) \in S_D$.

Remark that dual simulation enhances graph simulation by imposing the condition (c) in order to preserve both child and parent relationships. As mentioned in [13], the graph pattern matching via graph simulation (resp. dual simulation) is to find the the maximum match relation $S$ (resp. $S_D$). Ma et al. [14] show that graph/dual simulation may do excessive matching of pattern graphs which makes the graph result very large and difficult to understand and analysis. For this reason, they propose strong simulation, an extension of dual simulation by imposing the notion of locality. This notion requires that each subgraph of the final match result must have a radius bounded by the diameter of the pattern graph.

**D)** *Strong simulation:* $G$ matches $Q$ via strong simulation, denoted $Q \prec_D^p G$, if there exists a node $v \in V$ and a subgraph $G_s$ of $G$ centered at $v$ such that:
1. The radius of $G_s$ is bounded by $d_Q$, i.e., for each node $v' \in G_s$, $\text{dist}(v, v') \leq d_Q$;
2. $Q \prec_D G_s$ with the maximum match relation $S_D$.

Informally, rather than matching the whole data graph $G$ over $Q$ we extract, for each node $n \in V$, a subgraph $G_s$ of $G$ centered at $n$ and which has a radius equals to $d_Q$. Then, we match $G_s$ over $Q$ via dual simulation. In this way, the match result will be composed of subgraphs of reasonable size that satisfy both child and parent relationships of $Q$.

**Match results.**

**A)** When $Q \sim_{sio} G$ then the match result $M_{sio}(Q, G)$ is the set of all subgraphs of $G$ that are isomorphic to $Q$. **B)** When $Q \prec G$ with the maximum match relation $S$ then the match result $M(Q, G)$ w.r.t. $S$ is each subgraph $G(V_s, E_s)$ of $G$ in which: 1) a node $n \in V_s$ if it is in $S$; and 2) an edge $(v, v') \in E_s$ iff there exists an edge $(u, u') \in E_Q$ with $(u, v) \in S$ and $(u', v') \in S$. **C)** When $Q \prec_D G$ then the match result $M_D(Q, G)$ is defined similarly to graph simulation but w.r.t. the maximum match relation $S_D$. **D)** When $Q \prec_D^p G$ then the match result $M_D^p(Q, G)$ is defined with $\bigcup_i M_D(Q, G_i)$ where each $G_i$ is a subgraph of $G$ that satisfies the conditions of strong simulation.

**Potential matches.** Given a data graph $G(V, E, \lambda)$ and a pattern graph $Q(V_Q, E_Q, \lambda_Q)$. For any node $u \in V_Q$, we call potential match each node $v \in V$ that has the same label as $u$ (i.e. $\lambda_Q(u) = \lambda(v)$). Moreover, $\text{Sim}(u)$ refers to the set of all potential matches of $u$ in $G$. 

---

**XX:4** Graph Pattern Matching Preserving Label-Repetition Constraints
Example 2. Consider the data graph $G$ and the pattern graph $Q_2$ of Fig. 1. With dual simulation, both $G_1$ and $G_2$ are found as matches of $Q_2$ in $G$. Remark that the cycle of two nodes $A_1$ and $D_2$ in $Q_2$ is matched with the long cycle $d_9 \rightarrow \cdots \rightarrow d_{12} \rightarrow d_9$ in $G_2$, which may be hard to analysis. With the notion of locality, strong simulation returns $G_1$ as the only match of $G$ over $Q_2$ and ignores $G_2$ since it represents an excessive matching.

3 Triple Simulation

We start first by presenting a new topological constraint that one would like to preserve during graph pattern matching. We then define a new extension of graph simulation by imposing this constraint. We compare our extension with only strong simulation [14] since this is the more expressive graph pattern matching approach that requires a polynomial-time. Notice that another polynomial-time approach exists [7], called bounded simulation, which imposes constraints on edges. However, our extension concerns nodes constraints.

Given a data graph $G$ and consider the pattern graphs $Q_1 = a \rightarrow b$ and $Q_2 = b \leftarrow a \rightarrow b$. It is obvious that these two patterns are not equivalent: $Q_1$ requires that each node $v$ in $G$ that matches $a$ must have at least one child node labeled with $b$, however, $Q_2$ requires that $v$ must have at least two child nodes labeled with $b$. Strong simulation fails to make this difference and considers $Q_1$ and $Q_2$ as equivalent patterns (as illustrated by Example 1).

Definition 1. Given a data graph $G(V,E)$ and a pattern graph $Q(V_Q,E_Q)$. A Label-Repetition (LR) constraint defined over a node $u \in V_Q$ with label $l$ specifies that: 1) there is a maximum subset $C_u = \{u_1, \ldots, u_K\}$ ($K \geq 2$) of children (resp. parents) of $u$ that are all labeled with $l$; and 2) any match $v$ of $u$ in $G$ must have a subset $C_v = \{v_1, \ldots, v_K\}$ of children (resp. parents) ordered in such a way that allows to match each $v_i$ to a child $u_i$ of $u$.

Intuitively, a LR constraint concerns a repetition of some label either among children or among parents of some node in $Q$. If children (resp. parents) of each node in $Q$ have distinct labels, then $Q$ is defined with only child and parent relationships and, thus, can be matched correctly via strong simulation. The limitation of this latter is observed when some children (resp. parents) of the same node are defined with the same label.

Example 3. Consider the pattern graph $Q_1$ of Fig. 1. There is an LR constraint defined over the node $q_2$ with label $SE$. It specifies that each node of the data graph that matches $q_2$ must have at least two children labeled $SE$ such that one of them matches the node $q_3$ and the other one matches the node $q_4$.

We propose next a new extension of graph simulation in order to satisfy LR constraints.

Definition 2. A data graph $G(V,E,\lambda)$ matches a pattern graph $Q(V_Q,E_Q,\lambda_Q)$ via triple simulation, denoted by $Q \lessdot_T G$, if there exists a binary match relation $S_T \subseteq V_Q \times V$ s.t.:

1. For each $(u,v) \in S_T$, $\lambda_Q(u) = \lambda(v)$.
2. For each $u \in V_Q$ there exists $(u,v) \in S_T$.
3. For each $(u,v) \in S_T$ and for all edges $(u,u_1), \ldots, (u,u_n) \in E_Q$, there exists at least $n$ distinct children $v_1, \ldots, v_n$ of $v$ in $G$ such that: $(u_1, v_1), \ldots, (u_n, v_n) \in S_T$.
4. For each $(u,v) \in S_T$ and for all edges $(u_1, u), \ldots, (u_n, u) \in E_Q$, there exists at least $n$ distinct parents $v_1, \ldots, v_n$ of $v$ in $G$ such that: $(u_1, v_1), \ldots, (u_n, v_n) \in S_T$.

$M_T(Q,G)$ is the match result that corresponds to the maximum match relation $S_T$.\footnote{This match result can be defined similarly to graph (dual) simulation.}
Intuitively, if a node \( u \) in \( Q \) has \( n \) children (resp. parents) then each match \( v \) of \( u \) in \( G \) must have at least \( n \) distinct children (resp. parents) such that we can match, w.r.t some order, each child (resp. parent) of \( v \) to only one child (resp. parent) of \( u \). This new restriction imposed by conditions (3) and (4) prevents matching of distinct children (resp. parents) of some node \( u \) in \( Q \) to the same node in \( G \), as may be done by strong simulation. Notice that triple simulation preserves also child and parent relationships and not only LR constraints.

Example 4. Consider the data graph \( G \) and the pattern graphs \( Q_1 \) and \( Q_2 \) of Fig. 1. The node \( q_1 \) with label BIO in \( Q_1 \) has two parents, \( q_3 \) and \( q_4 \), that have the same label SE. Remark that \( d_1 \) and \( d_2 \) are potential matches of \( q_1 \) in \( G \). According to triple simulation, \( d_1 \) (resp. \( d_2 \)) must have at least two distinct parents s.t. one can match \( q_3 \) and the other one can match \( q_4 \). This is not the case since \( d_1 \) (resp. \( d_2 \)) has only one parent labeled SE. Thus, we can conclude that no subgraph in \( G \) satisfies the LR constraint of \( Q_1 \), and then, \( M_T(Q_1, G) = \emptyset \). When triple simulation is adopted for \( Q_2 \) over the subgraph \( G_2 \), we obtain the following maximum match relation: \( S_T = \{(q_1, d_1), (q_2, d_3), (q_4, d_4), (q_5, d_5), (q_6, d_6)\} \). The match result that corresponds to \( S_T \) is the whole subgraph \( G_2 \), which is correct.

We use CPL relationships to refer to Child and Parent relationships (called duality properties), as well as relationships based on LR constraints. Our motivation is to propose a graph pattern matching algorithm that preserves CPL relationships in polynomial-time.

4 Satisfy LR Constraints

We first present the problem of satisfying LR constraints and show that a naive approach may lead for exponential cost. Next, we define a condition that is necessary and sufficient for the satisfaction of LR constraints and which can be checked in polynomial-time.

Example 5. Consider the graphs depicted in Fig. 2. The pattern graph \( Q \) looks for each professor (Pr) which has supervised at least three PhD thesis in topics related respectively to Cloud Computing (CC), Collaborative Editing (CE) and Electronic Vote (EV). The node \( d_1 \) in \( G_1 \) is a potential match of \( q_1 \). To satisfy the condition (3) of triple simulation (Definition 3), \( d_1 \) must have at least three child nodes which is the case, and there must be some order that allows to match each child of \( d_1 \) to a child of \( q_1 \). However remark that: if we match \( q_2 \) with \( d_2 \) then we can not have match neither for \( q_3 \) nor for \( q_4 \); and moreover, if we match \( q_2 \) with \( d_3 \) then we can match either \( q_3 \) with \( d_2 \) or \( q_4 \) with \( d_2 \). Clearly, there is no order over the children \( d_2, d_3, d_4 \) of \( d_1 \) that allows to match all the children \( q_2, q_3, q_4 \) of \( q_1 \) in \( Q \). Therefore, the data graph \( G_1 \) does not satisfy the LR constraint of \( Q \). On the other side, the data graph \( G_2 \) match correctly \( Q \): see that there is an order that allows to match each child of \( d_1 \) to a child of \( q_1 \), i.e., \( q_2, q_3, q_4 \) can be matched respectively with \( d_3, d_4, d_2 \). Thus, the LR constraint of \( Q \) is satisfied over \( G_2 \).
Given the above, one can think that checking LR constraints may lead to exponential cost (since we must consider all orders over some data nodes). However, we show later that this process can be done in polynomial-time.

**Definition 3.** Given a data graph \( G(V, E) \) and a pattern graph \( Q(V_Q, E_Q) \). Consider all the LR constraints defined over children (resp. parents) of some node \( u \in V_Q \), and let \( v \in V \) be a potential match of \( u \). The bipartite graph \( BG(X \cup Y, E) \) that inspects these LR constraints defined over children (resp. parents) of \( u \) is defined as follows:

- \( X \subseteq V_Q \) contains each child (resp. parent) of \( u \) that is concerned by an LR constraint.
- \( Y \subseteq V \) contains each child (resp. parent) of \( v \) that (potentially) matches some node in \( X \).
- \((u', v') \in E \) if \( u' \in X \) is (potentially) matched with \( v' \in Y \).

A complete matching over \( BG \) is a maximum matching \( \downarrow \) that covers each node in \( X \) \( \uparrow \).

Consider only the LR constraints defined over children of \( u \). The set \( X \) of the bipartite graph \( BG \) contains all children of \( u \) that are concerned by some LR constraint, and the set \( Y \) contains each child of \( v \) (potentially) matches some child \( u' \) of \( u \), provided that \( u \) is concerned by an LR constraint (i.e. \( u' \in X \)). Moreover, an edge in \( E \subseteq X \times Y \) denotes some child of \( u \) in \( X \) that can be (potentially) matched with some child of \( v \) in \( Y \). For LR constraints defined over parents of \( u \), the bipartite graph that inspects them is defined in the same manner (i.e. \( X \) is a subset of parents of \( u \), and \( Y \) is a subset of parents of \( v \)).

**Example 6.** Consider the pattern graph \( Q \) and data graphs \( G_1 \) and \( G_2 \) depicted in Fig. 2. Recall that there is an LR constraint defined over the children of the node \( q_1 \) in \( Q \). The bipartite graph \( BG_1 \) that inspects this LR constraint, w.r.t the potential match \( d_1 \) of \( q_1 \) in \( G_1 \), is depicted in Fig. 2(d). Moreover, w.r.t the potential match \( d_1 \) of \( q_1 \) in \( G_2 \), the corresponding bipartite graph \( BG_2 \) is given in Fig. 2(e).

The next theorem states our main contribution which is a necessary and sufficient condition to satisfy LR constraints.

**Theorem 1.** Given a data graph \( G(V, E) \), a pattern graph \( Q(V_Q, E_Q) \), and a node \( u \in V_Q \) with a potential match \( v \in V \). Let \( BG \) be the bipartite graph that inspects all the LR constraints defined over children (resp. parents) of \( u \) w.r.t \( v \). These LR constraints are satisfied by some children (resp. parents) of \( v \) iff there is a complete matching over \( BG \). Moreover, this can be decided in at most \( O(|V_Q||V|\sqrt{|V_Q|} + |V|) \) time.

We emphasize that for each node \( u \) in \( Q \) and each potential match \( v \) of \( u \) in \( G \), we construct at most two bipartite graphs, the first one to inspect LR constraints that are defined over children of \( u \), and the second one to inspect those defined over parents of \( u \).

**Example 7.** As explained in Example 6, the LR constraint defined over the children of \( q_1 \) in \( Q \) is not satisfied by the children of its potential match \( d_1 \) in \( G_1 \). This is confirmed by the bipartite graph \( BG_1 \) of Fig. 2(d) which has a maximum matching of size 2 (does not cover the set \( X \)). Thus, no complete matching exists over \( BG_1 \) and, according to Theorem 1, we can conclude that the underlying LR constraint is not satisfied by the children of \( d_1 \). Consider the bipartite graph \( BG_2 \) of Fig. 2(e) that inspects the same LR constraint w.r.t the potential match \( d_1 \) of \( G_2 \). Bold edges in \( BG_2 \) represent a maximum matching of size 3. Thus, a complete matching exists over \( BG_2 \) which implies that the LR constraint, defined over the children of \( q_1 \) in \( Q \), is satisfied by the children of its potential match \( d_1 \) of \( G_2 \).

---

\(^3\) It is also called X-saturating matching.
5 An Algorithm for Triple Simulation

Our algorithm, referred to as TSim, is shown in the Fig. [3]. Given a pattern graph $Q$ and a data graph $G$, TSim($Q, G$) returns the match result $M_T(Q, G)$, if $Q \prec_T G$, and $\emptyset$ otherwise. This match result contains each subgraph of $G$ that satisfies all CPL relationships of $Q$.

First, we compute for each node $u \in V_Q$, the set $\text{SIM}(u)$ of all its potential matches in $V$ [lines 1-3]. In order to preserve efficiently the CPL relationships of $Q$ over $G$, we define four auxiliary structures [line 4] as follows. For any node $u \in V_Q$, $\text{CP}(Q, u)$ contains all children and parents of $u$ that are concerned by Child and/or Parent relationships; and $\text{LR}(Q, u)$ contains those concerned by some LR constraints. Moreover, for each potential match $v$ of $u$ in $G$, $\text{ChildAsMatch}(Q, G, v, u)$ returns the number of $v$’s children that are potential matches of $u$ in $G$ (i.e. each child $v'$ of $v$ with $v' \in \text{SIM}(u)$); and $\text{ParentAsMatch}(Q, G, v, u)$ returns the number of $v$’s parents that are potential matches of $u$ in $G$.

Algorithm TSim preserves the Child and Parent relationships of $Q$ [lines 6-15] as follows. Given a node $u \in V_Q$, a potential match $v$ of $u$ is kept in $\text{SIM}(u)$ unless: 1) $u$ has a child $v' \in \text{CP}(Q, u)$ but $v$ has no child that matches $u'$ (i.e. $\text{ChildAsMatch}(Q, G, v, u')=0$); or 2) $u$ has a parent $u' \in \text{CP}(Q, u)$ but $v$ has no parent that matches $u'$ (i.e. $\text{ParentAsMatch}(Q, G, v, u')=0$). If one of these two conditions is satisfied then $v$ is an incorrect match of $u$, w.r.t duality properties, and is removed from $\text{SIM}(u)$ [lines 8 + 13]. The checking of LR constraints [lines 17-19] is done through the procedure LRChecking. Given a node $u \in V_Q$ with a potential match $v \in V$. According to Definition [3] the procedure LRChecking constructs two bipartite graphs: $BG_1$ that inspects all the LR constraints defined over the children of $u$ [lines 2-5]; and $BG_2$ that inspects those defined over the parents of $u$ [lines 6-9]. If a complete matching exists over $BG_1$ and another one exists over $BG_2$ then, according to Theorem [1] we conclude that: a) all the LR constraints defined over the children of $u$ are satisfied by some children of $v$; and b) all the LR constraints defined over the parents of $u$ are satisfied by some parents of $v$. Thus, the procedure returns true only if these two complete matching exist over $BG_1$ and $BG_2$. If the procedure returns false then there is at least one LR constraint defined over the children (resp. parents) of $u$ which is not satisfied by the children (resp. parents) of $v$. In this case, $v$ is an incorrect match of $u$, w.r.t LR constraints, and is removed from $\text{SIM}(u)$ [line 18]. The procedure CompleteMatch [5] is an implementation of the algorithm of Hopcroft and Karp [13].

Each time a data node $v$ is removed from $\text{SIM}(u)$, the cardinalities stored by the structures ChildAsMatch and ParentAsMatch are updated according to the couple $(u, v)$. This is done by the procedure UpdateStruct. The two phases discussed above (checking of duality properties and LR constraints) are repeated by algorithm TSim until there are no more changes [lines 5-22]. Finally, the maximum match relation $M_T$ that corresponds to Definition [5] is defined, and its corresponding match result $M_T(Q, G)$ is constructed and returned.

Theorem 2. For any pattern graph $Q(V_Q, E_Q)$ and data graph $G(V, E)$, algorithm TSim takes at most $O(|Q||G| + |V_Q|^3|V|^2 \sqrt{|V_Q|} + |V|)$ time to decide whether $Q \prec_T G$ and to find the match result $M_T(Q, G)$. Moreover, it takes $O(|Q||G|)$ time if $Q$ has no LR constraint [5].

The worst-case time complexity of TSim is bounded by $O(|Q|^2|G|^{1.5})$. As opposed to the NP-Completeness of its traditional counterpart via subgraph isomorphism [13], triple simulation allows to match pattern graphs with LR constraints in polynomial-time.

---

4 This procedure finds the maximum matching over $BG_1$ (resp. $BG_2$), using the algorithm of Hopcroft et al. [13], and then checks whether the size of this maximum matching is equals to $|X_1|$ (resp. $|X_2|$).

5 Given a graph $G(V, E)$, $|G| = |V| + |E|$. 
Algorithm TSim\((Q, G)\)
\(\textbf{Input:}\) Graph pattern \(Q(V_Q, E_Q, \lambda_Q)\), data graph \(G(V, E, \lambda)\).
\(\textbf{Output:}\) The match result \(\mathcal{M}_T(Q, G)\) if \(Q \prec_T G\) and \(\emptyset\) otherwise.

1: \textbf{for each} \(u \in V_Q\) \textbf{do} /* \textsc{Potential matches of each node in} \(Q\) */
2: \hspace{1em} \(\text{SIM}(u) := \{v \mid v \in V \text{ and } \lambda_Q(u) = \lambda(v)\}\)
3: \textbf{end for}
4: \text{initAuxStruct}(Q, G);
5: \textbf{do}
6: \hspace{1em} \textbf{for each} \((u, v)\) \textbf{with} \(v \in \text{SIM}(u)\) \textbf{do}
7: \hspace{2em} \textbf{for each} child \(u'\) \textbf{of} \(u\) \textbf{with} \(u' \in \text{CP}(Q, u)\) \textbf{do} /* \textsc{Preserving Child relations} */
8: \hspace{3em} \textbf{if} \(\text{ChildAsMatch}(Q, G, v, u') = 0\) \textbf{then}
9: \hspace{4em} \(\text{SIM}(u) := \text{SIM}(u) \setminus \{v\}\); \text{UpdateStruct}(G, u, v);
10: \hspace{3em} \textbf{end if}
11: \hspace{2em} \textbf{end for}
12: \hspace{1em} \textbf{for each} parent \(u'\) \textbf{of} \(u\) \textbf{with} \(u' \in \text{CP}(Q, u)\) \textbf{do} /* \textsc{Preserving Parent relations} */
13: \hspace{2em} \textbf{if} \(\text{ParentAsMatch}(Q, G, v, u') = 0\) \textbf{then}
14: \hspace{3em} \(\text{SIM}(u) := \text{SIM}(u) \setminus \{v\}\); \text{UpdateStruct}(G, u, v);
15: \hspace{3em} \textbf{end if}
16: \hspace{1em} \textbf{end for}
17: \hspace{1em} \textbf{if} \(\text{LR\_Checking}(Q, G, v, u) = \text{false}\) \textbf{then} /* \textsc{Preserving LR constraints} */
18: \hspace{2em} \(\text{SIM}(u) := \text{SIM}(u) \setminus \{v\}\); \text{UpdateStruct}(G, u, v);
19: \hspace{2em} \textbf{end if}
20: \hspace{1em} \textbf{end if}
21: \hspace{1em} \textbf{end for}
22: \textbf{while} there are changes;
23: \(S_T := \{(u, v) \mid u \in V_Q \text{ and } v \in \text{SIM}(u)\}\);
24: \textbf{Construct the match result} \(\mathcal{M}_T(Q, G)\) that corresponds to \(S_T\);
25: \textbf{return} \(\mathcal{M}_T(Q, G)\);

\textbf{Procedure} UpdateStruct\((Q, G, u, v)\)
\(\textbf{Input:}\) A pattern graph \(Q\), data graph \(G(V, E)\), a query node \(u\) with a removed potential match \(v\).
\(\textbf{Output:}\) Updates the auxiliary structures \text{ChildAsMatch} and \text{ParentAsMatch}.

1: \textbf{Do ChildAsMatch}(Q, G, v', u) := ChildAsMatch(Q, G, v', u) - 1 \textbf{for each} \((v', v) \in E)\;
2: \textbf{Do ParentAsMatch}(Q, G, v', u) := ParentAsMatch(Q, G, v', u) - 1 \textbf{for each} \((v, v') \in E)\;

\textbf{Procedure} LR\_Checking\((Q, G, u, v)\)
\(\textbf{Input:}\) Graph pattern \(Q(V_Q, E_Q)\), data graph \(G(V, E)\), a node \(u \in V_Q\) with a potential match \(v \in V\).
\(\textbf{Output:}\) Whether all the LR constraints defined over \(u\) are satisfied by children and/or parents of \(u\).

1: \(BG_1 := (X_1 \cup Y_1, E_1); BG_2 := (X_2 \cup Y_2, E_2)\); where \(X_1 = Y_1 = X_2 = Y_2 = E_1 = E_2 = \emptyset\);
2: \textbf{for each} child \(u'\) \textbf{of} \(u\) \textbf{with} \(u' \in \text{LR}(Q, u)\) \textbf{do}
3: \hspace{1em} \(X_1 := X_1 \cup \{u'\}\);
4: \hspace{1em} \textbf{Do} \(Y_1 := Y_1 \cup \{v'\}\); \(E_1 := E_1 \cup \{(u', v')\}\); \textbf{for each} \((v' \in \text{SIM}(u')\) \textbf{with} \((v', v) \in E)\);
5: \hspace{1em} \textbf{end for}
6: \hspace{1em} \textbf{for each} parent \(u'\) \textbf{of} \(u\) \textbf{with} \(u' \in \text{LR}(Q, u)\) \textbf{do}
7: \hspace{2em} \(X_2 := X_2 \cup \{u'\}\);
8: \hspace{2em} \textbf{Do} \(Y_2 := Y_2 \cup \{v'\}\); \(E_2 := E_2 \cup \{(u', v')\}\); \textbf{for each} \((v' \in \text{SIM}(u')\) \textbf{with} \((v', v) \in E)\);
9: \hspace{1em} \textbf{end for}
10: \textbf{return} \text{true} if \((\text{CompleteMatch}(BG_1) & \text{CompleteMatch}(BG_2))\); \text{false} otherwise;

Figure 3 Algorithm for Triple Simulation.
6 Triple Simulation with Locality

The next example suggests to incorporate the notion of locality [14] into our algorithm TSIM in order to overcome excessive matching and thus to improve the quality of our match results.

Example 8. Consider the graphs depicted in Fig. 1. We extend the subgraph $G_1$ with the following relationships: $d_1 \leftarrow d_{13} \leftarrow d_7$ where $d_{13}$ is a new node labeled with SE. Let $G'_1$ be the subgraph that results from this modification. When triple simulation is adopted, TSIM returns $G'_1$ as the only match of $Q_1$ in $G$. The BIO found in $G'_1$ (node $d_2$) is recommended by two SE ($d_8$ and $d_{13}$) as specified by $Q_1$. However, TSIM returns an excessive match of the cycle $AI \Rightarrow DM$, i.e. the cycle $d_9 \rightarrow \cdots \rightarrow d_{12} \rightarrow d_9$ in $G'_1$, that one does not want.

Next is a new definition of triple simulation that takes into account the notion of locality.

Definition 4. A data graph $G$ matches a pattern graph $Q$ via triple simulation and under locality, denoted $Q \prec_T^L G$, if there exists a subgraph $G_s$ of $G$ centered at some node $v$ s.t.: 1. the radius of $G_s$ is bounded by $d_Q$, i.e., for each node $v$ in $G_s$, $\text{dist}(v, v') \leq d_Q$; and 2. $Q \prec_T G_s$ with the maximum match relation $S_T$.

The match result $M(T, Q, G)$ is defined with $\bigcup_i M(T, Q, G_i)$ where each $G_i$ is a subgraph of $G$ that satisfies the previous conditions.

To implement the Definition 4 one can replace only the procedure dualSim in the algorithm Match [14] with our algorithm TSIM. Let $\text{Match}^+$ be the algorithm that results from this combination. Given a data graph $G$ and a pattern graph $Q$. Algorithm $\text{Match}^+$ extracts a subgraph $G_v$ over each node $v$ in $G$, provided that its radius does not exceed $d_Q$. It then matches $G_v$ over $Q$ via triple simulation (instead of dual simulation). The match found on each subgraph has a reasonable size and satisfies all the CPL relationships of $Q$.

Theorem 3. For any pattern graph $Q(V_Q, E_Q)$ and data graph $G(V, E)$, algorithm $\text{Match}^+$ takes at most $O(|V|^2 + |Q||G||V| + |V_Q|^3|V|^3 \sqrt{|V_Q| + |V|})$ time to decide whether $Q \prec_T^L G$ and to find the corresponding match result $M(T, Q, G)$.

The complexity of $\text{Match}^+$ is bounded by $O(|Q|^2|G|^2)$ while that of $\text{Match}[14]$ is bounded by $O(|Q||G|^2)$. This promises that combining our results with existing orthogonal approaches will not increase drastically the complexity of graph pattern matching.

7 Conclusion

We have discussed pattern graphs with LR constraints that existing approaches do not preserve [4, 7] or preserve in exponential time [10]. To tackle this NP-Completeness, we have showed that LR constraints can be preserved in polynomial-time when treated as maximum matching in bipartite graphs, and we proposed an algorithm to implement this result.

We are to study other constraints that can be preserved in polynomial-time, e.g., negation and optional edges. The polynomial-time of our algorithm may make graph pattern matching infeasible when conducted on graphs with millions of nodes and billions of edges (e.g. Facebook [11]). To boost the matching on large data graphs, we plan to extend our work with some optimization techniques: 1) incremental graph pattern matching [11], 2) pattern matching on distributed data graphs [2, 20, 13], and 3) pattern matching on compressed data graphs [8, 15]. These techniques are orthogonal, but complementary, to our work.

6 Not given here since its definition is trivial.
7 This result is a combination of Theorem 3 and Theorem 4.1 of Ma et al. [14].
References

1. Junghoo Cho, Narayanan Shivakumar, and Hector Garcia-Molina. Finding replicated web collections. In Proc. of SIGMOD, pages 355–366, 2000.
2. Gao Cong, Wenfei Fan, and Anastasios Kementsietsidis. Distributed query evaluation with performance guarantees. In Proc. of SIGMOD, pages 509–520, 2007.
3. Luigi P. Cordella, Pasquale Foggia, Carlo Sansone, and Mario Vento. A (sub)graph isomorphism algorithm for matching large graphs. IEEE Trans. Pattern Anal. Mach. Intell., pages 1367–1372, 2004.
4. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Introduction to Algorithms, Third Edition. The MIT Press, 3rd edition, 2009.
5. Wenfei Fan. Graph pattern matching revised for social network analysis. In Proc. of ICDT, pages 8–21, 2012.
6. Wenfei Fan, Jianzhong Li, Shuai Ma, Nan Tang, and Yinghui Wu. Adding regular expressions to graph reachability and pattern queries. In Proc. of ICDE, pages 39–50, 2011.
7. Wenfei Fan, Jianzhong Li, Shuai Ma, Nan Tang, Yinghui Wu, and Yumeng Wu. Graph pattern matching: From intractable to polynomial time. Proc. VLDB Endow., pages 264–275, 2010.
8. Wenfei Fan, Jianzhong Li, Xin Wang, and Yinghui Wu. Query preserving graph compression. In Proc. of SIGMOD, pages 157–168, 2012.
9. Wenfei Fan, Xin Wang, and Yinghui Wu. Incremental graph pattern matching. ACM Trans. Database Syst., pages 18:1–18:47, 2013.
10. Wenfei Fan, Yinghui Wu, and Jingbo Xu. Adding counting quantifiers to graph patterns. In Proc. of SIGMOD, pages 1215–1230, 2016.
11. Ivana Grujic, Sanja Bogdanovic Dinic, and Leonid Stoimenov. Collecting and analyzing data from e-government facebook pages. In Proceedings of ICT Innovations, pages 86–96, 2014.
12. P. Hall. On representatives of subsets. Journal of the London Mathematical Society, s1-10(1):26–30, 1935.
13. John E. Hopcroft and Richard M. Karp. An n5/2 algorithm for maximum matchings in bipartite graphs. SIAM J. Comput., 2(4):225–231, 1973.
14. Shuai Ma, Yang Cao, Wenfei Fan, Jinfeng Huai, and Tianyu Wo. Strong simulation: Capturing topology in graph pattern matching. ACM Trans. Database Syst., 39(1):4:1–4:46, 2014.
15. Antonio Maccioni and Daniel J. Abadi. Scalable pattern matching over compressed graphs via dedensification. In Proc. of SIGKDD, pages 1755–1764, 2016.
16. R. Milner. Communication and Concurrency. Prentice-Hall, Inc., 1989.
17. Robin Milner. Communication and concurrency. Prentice Hall, 1989.
18. Ali Shemshadi, Quan Z. Sheng, and Yongrui Qin. Efficient pattern matching for graphs with multi-labeled nodes. Knowl.-Based Syst., 109:256–265, 2016.
19. Le-Duc Tung, Quyet Nguyen-Van, and Zhenjiang Hu. Efficient query evaluation on distributed graphs with hadoop environment. In Proc. of SoICT, pages 311–319, 2013.
20. Xin Wang, Junlu Ji, and Xiaowang Zhang. Efficient distributed regular path queries on RDF graphs using partial evaluation. In Proc. of CIKM, pages 1933–1936, 2016.
XX:12 Graph Pattern Matching Preserving Label-Repetition Constraints

APPENDIX

A Proof of Theorem 1

Theorem 1. (Recall) Given a data graph \( G(V, E) \), a pattern graph \( Q(V_Q, E_Q) \), and a node \( u \in V_Q \) with a potential match \( v \in V \). Let \( BG \) be the bipartite graph that inspects all the LR constraints defined over children (resp. parents) of \( u \) w.r.t \( v \). These LR constraints are satisfied by some children (resp. parents) of \( v \) iff there is a complete matching over \( BG \). Moreover, this can be decided in at most \( O(|V_Q||V|\sqrt{|V_Q|} + |V|) \) time.

To simplify the proof, we consider only the case of LR constraints defined over children of \( u \). The second case, i.e. when parents of \( u \) are concerned by some LR constraints, can be studied in the same way. Satisfying LR constraints is closer to the problem of perfect matching in bipartite graph \( [1] \), or moreover, a System of Distinct Representatives \( [12] \). In our case, node sets \( X \) and \( Y \) of our bipartite graphs have not the same size then we use the term of complete matching instead of perfect matching. Given a bipartite graph \( B=(X \cup Y, Z) \). A maximum matching \( S \subseteq Z \) is the largest subset of the edge set \( Z \) such that no two edges start/end at the same node. If \( S \) is a complete matching, i.e. \( |S| = |X| \), then for each node \( x \in X \) there is one and only one edge \((x,y) \in S\) that connects it with a node \( y \in Y \). We say that all elements of \( X \) are covered (i.e. matched).

\[ \implies \] Consider that all LR constraints defined over children of \( u \) are satisfied by some children of its potential match \( v \). Recall that \( BG \) is defined with \((X \cup Y, Z)\) where \( X \) contains each child of \( u \) that is concerned by an LR constraint; and \( Y \) contains each child of \( v \) that matches at least one child of \( u \) in \( X \). Let \( K \) be the number of \( u \)’s children that are concerned by LR constraints (i.e. \( K = |X| \)). Since all the LR constraints in question are satisfied by some children of \( v \) then, for each single one defined over the subset \( C_u = u_1, \ldots, u_N \) of \( N \) children of \( u \) \((2 \leq N \leq K)\), \( v \) satisfies condition (2) of Definition \( [1] \) and has a subset \( C_v = v_1, \ldots, v_N \) of children such that each \( v_i \) matches a child \( u_i \) of \( u \). Notice that two different LR constraints are defined with two different labels, thus children of \( v \) that satisfy one LR constraint are different from those that satisfy another LR constraint. By following the same principle, to satisfy all LR constraints defined over children of \( u \), \( v \) has certainly \( K \) distinct children such that each one is matched to only one child of \( u \) which is concerned by some LR constraint. This matching can be represented by \( K \) edges that connect each child of \( v \) in \( Y \) to only one child of \( u \) in \( X \) \((*)\). Moreover, if two children of \( v \) has the same label then they are concerned by the same LR constraint and, according to Definition \( [1] \) are matched to different nodes in \( X \) \((**)\). From \((*)\) and \((**)\), we conclude that these \( K \) edges do not start/end at the same node and then represent a complete matching over the bipartite graph \( BG \). Therefore, if all LR constraints defined over children of \( u \) are satisfied by some children of \( v \), then there is a complete matching over the bipartite graph \( BG \) that inspects these LR constraints w.r.t \( v \).

\[ \iff \] Consider that there is a complete matching over the bipartite graph \( BG \). According to our definition of complete matching, there is an edge that connects each node in \( X \) (i.e. a child \( u’ \) of \( u \) that is concerned by an LR constraint) to only one node in \( Y \) (i.e. a child \( v’ \) of \( v \) with \( v’ \in \text{sim}(u’) \)), and moreover, each node in \( Y \) is connected to only one node in \( X \). We conclude that \( v \) has at least \( K \) children \((K = |X|)\) and there exists an order over these children that allows to match each one to only one child of \( u \) which is concerned by some LR constraint. Therefore, according to Definition \( [1] \) each LR constraint defined over some children of \( u \) is satisfied by some children of \( v \). The node set \( X \) (resp. \( Y \)) of the bipartite graph \( BG \) may have at most \(|V_Q|\) (resp. \(|V|\))
nodes. Moreover, the edge set \( Z \) may have at most \( |V_Q||V| \) edges. To check whether there exists a complete matching over \( BG \), we look first for the maximum matching over \( BG \) and we then check whether its cardinality is equal to \( |X| \). The best algorithm to find a maximum matching over a bipartite graph with node set \( N \) and edge set \( M \), discovered by Hopcroft and Karp [13], runs in \( O(|M|\sqrt{|N|}) \) time. Thus, by using this algorithm, the necessary and sufficient condition of Theorem 1 can be checked in at most \( O(|V_Q||\sqrt{|V_Q|}+|V|) \) time.

B Proof of Theorem 2.

Theorem 2. (Recall) For any pattern graph \( Q(V_Q, E_Q) \) and data graph \( G(V, E) \), algorithm \( TSim \) takes at most \( O(|Q||G| + |V_Q|^2\sqrt{|V_Q|} + |V|) \) time to decide whether \( Q \prec_T G \) and to find the match result \( M_T(Q, G) \). Moreover, it takes \( (|Q||G|) \) time in the absence of \( LR \) constraints.

Given a pattern graph \( Q(V_Q, E_Q, \lambda_Q) \) and a data graph \( Q(V, E, \lambda) \). It takes \( O(|V_Q||V|) \) time to compute \( \text{sim} \) sets for all query nodes of \( Q \) [lines 1-3]. We define each \( \text{sim}(u) \) as an indexed structure which allows, in constant time, 1) to check whether some data node \( v \) belongs to \( \text{sim}(u) \); or 2) to remove it from \( \text{sim}(u) \).

(A) The auxiliary structures \( \text{CP} \) and \( \text{LR} \) can be constructed in at most \( O(|V_Q|^2) \) time as follows. For any node \( u \in V_Q \), we define an indexed list \( \text{LabelOcc}(u, l) \) which returns the number of children of \( u \) that are labeled with \( l \). This list can be constructed in \( O(|V_Q|) \) time by parsing all children of \( u \). For each child \( u' \) of \( u \), if \( \text{LabelOcc}(u, \lambda_Q(u')) > 1 \), then other children of \( u \) have the same label as \( u' \). Thus, \( u' \) is concerned by an \( LR \) constraint and must belong to \( \text{LR}(Q, u) \). Otherwise, \( \text{LabelOcc}(u, \lambda_Q(u')) = 1 \), \( u' \) is the unique child of \( u \) that has the label \( \lambda_Q(u') \) and thus must belong to \( \text{CP}(Q, u) \). This process is repeated similarly over parents of \( u \) to complete the definition of \( \text{CP}(Q, u) \) and \( \text{LR}(Q, u) \). It is clear that for each node \( u \in V_Q \), \( \text{CP}(Q, u) \) and \( \text{LR}(Q, u) \) can be constructed in \( O(|V_Q|) \) time. Therefore, for all nodes of \( Q \), the cost becomes \( O(|V_Q|^2) \).

(B) It is easy to verify that for each query node \( u \in V_Q \) and data node \( v \in V \), \( \text{ChildAsMatch}(Q, G, v, u) \) (resp. \( \text{ParentAsMatch}(Q, G, v, u) \)) can be constructed in \( O(|V|) \) time by parsing each child (resp. parent) of \( v \) and checking, in constant time, if this child belongs to \( \text{sim}(u) \). Therefore, by considering all nodes of \( Q \) and \( G \), the structures \( \text{ChildAsMatch} \) and \( \text{ParentAsMatch} \) can be constructed in at most \( O(|V_Q||V|^2) \) time.

(C) In addition to the four auxiliary structures described above, we construct in \( O(|E|) \) time (resp. \( O(|E_Q|) \) time) an indexed structure over the edges of \( E \) (resp. \( E_Q \)) in order to check in constant time whether some data edge (resp. query edge) exists. Moreover, we define sets of children and parents of each query node \( u \in V_Q \) (resp. data node \( v \in V \)) which can be done in \( O(|E_Q|) \) time (resp. \( O(|E|) \) time).

From (A), (B) and (C), we conclude that the cost of the call \( \text{initAuxStruct}(Q, G) \) [line 4] remains bounded by \( O(|V_Q||V|^2) \).

Each time we remove some data node \( v \) from \( \text{sim}(u) \), the procedure \( \text{UpdateStruct}(u, v) \) of Fig. 3 takes \( O(|V|) \) time to update the structures \( \text{ChildAsMatch} \) and \( \text{ParentAsMatch} \). This remove operation can be done at most \( |V_Q||V| \) time. Thus, the lines [9+14+18] of algorithm \( TSim \) take at most \( O(|V_Q||V|^2) \) time.

Given a query node \( u \) with a potential match \( v \). The checking of \( \text{Child} \) relationships [lines
Graph Pattern Matching Preserving Label-Repetition Constraints

Procedure MatchResult\(\mathcal{M}_T(Q, G, S_T)\)

Input: A pattern graph \(Q(V_Q, E_Q, \lambda_Q)\), a data graph \(G(V, E, \lambda)\), and the maximum match relation \(S_T\) for which \(Q \sim_T G\).

Output: The match result \(\mathcal{M}_T(Q, G)\) that corresponds to \(S_T\).

1: \(\mathcal{M}_T(Q, G) := (V_r, E_r, \lambda_r)\);
2: for each \((u, v) \in S_T\) do
3: \(V_r := V_r \cup \{v\}; \lambda_r(v) := \lambda(v)\);
4: end for
5: for each edge \((u, u') \in E_Q\) do
6: for each \((u, v) \in S_T\) and \((u', v') \in S_T\) do
7: if \((v, v') \in E\) then \(E_r := E_r \cup \{(v, v')\}\); end if
8: end for
9: end for
10: return \(\mathcal{M}_T(Q, G)\);

\[\text{Figure 4} \quad \text{Procedure to construct Match Results.}\]

7-11], as well as \(Parent\) relationships [lines 12-16] is done in at most \(O(|V_Q|)\) time by using the indexed structures \texttt{ChildAsMatch} and \texttt{ParentAsMatch} (inspired from [16]). Recall that the cost necessary to update these indexed structures is computed separately.

The call \texttt{LR\_Checking}(\(Q, G, u, v\)) [line 17] is done in at most \(O(|V_Q| V \sqrt{|V_Q| + |V|})\) time as we explain hereafter. As depicted by the procedure \texttt{LR\_Checking} of Fig. [3] we construct first two bipartite graphs \(BG_1\) and \(BG_2\) that inspect the \(LR\) constraints defined over children of \(u\) [lines 2-5] and those defined over parents of \(u\) respectively [lines 6-9]. We get all children/parents of \(u\) in at most \(O(|V_Q|)\) time by using our precomputed sets of children and parents. Thus, the construction of \(BG_1\) as well as \(BG_2\) requires a time bounded by \(O(|V_Q| V)\). Next, we use the procedure \texttt{CompleteMatch} (not detailed here) to check whether there exist two complete matchings over \(BG_1\) and \(BG_2\) respectively. Our bipartite graphs have at most \(|V_Q| V|E_Q| + |V|\) nodes and \(|V_Q| V|E_Q|\) edges. According to Theorem [1] the existence of complete matching over \(BG_1\) and \(BG_2\) can be checked in at most \(O(|V_Q| V \sqrt{|V_Q| + |V|})\) time. Therefore, the checking of \(LR\) constraints by algorithm \texttt{TSim} [lines 17-19] requires a time bounded by \(O(|V_Q| V \sqrt{|V_Q| + |V|})\).

For a query node \(u\) with a potential match \(v\), the checking of duality properties takes \(O(|V_Q|)\) time while that of \(LR\) constraints takes \(O(|V_Q| V \sqrt{|V_Q| + |V|})\) time. This tells us that the worst case arises when children (resp. parents) of \(u\) are concerned by only \(LR\) constraints. The checking process [lines 6-21] over all potential matches of \(u\) is done in at most \(O(|V_Q| V)\) time, in case of duality properties only, and in \(O(|V_Q| V^2 \sqrt{|V_Q| + |V|})\) time in case of \(LR\) constraints only.

Inspired from [10], the checking process (of duality properties and \(LR\) constraints) [lines 5-22] is executed over the nodes of \(Q\) in a deterministic manner: first over a randomly-chosen query node \(u\), after over adjacent nodes of \(u\) (children and parents) and so on. In this way, each time some \texttt{sim} set is changed we repeat the checking process over all already visited nodes since this change may influence on their \texttt{sim} sets. Thus, the \texttt{Do-While} loop will repeat the checking process \(|V_Q|\) times over each query node in \(Q\).

The definition of the maximum match relation \(S_T\) [line 23] can be done in at most \(O(|V_Q| V)\) time. The match result \(\mathcal{M}_T(Q, G)\) that corresponds to \(S_T\) can be defined in at most \(O(|E_Q| E|)\) time [line 24]. To prove this cost, we give in Fig. [4] the procedure \texttt{MatchRe-}
**Theorem 3.** (Recall) Authors of [10] propose a new extension of subgraph isomorphism by supporting simple counting quantifiers (CQs) on edges. These CQs can express universal and existential quantification, numeric and ratio aggregate, as well as negation.

**Example 1.** The pattern graph $Pr \geq_{100\%} PhD \rightarrow Conf\_Paper$ looks for each professor such that all her PhD students (universal quantification) have at least one conference paper (existential quantification). The pattern graph $Pr \geq_{60\%} PhD \rightarrow Conf\_Paper \rightarrow DBLP$ looks
for each professor such that 60% of her PhD students (aggregate ratio) have at least two conference papers (numeric ratio) that are not indexed in DBLP (negation).

Definition 2. A pattern graph with counting quantifiers, called quantified pattern graph (QGP), is defined with $Q(V,E,\lambda,C)$ where $V, E, \lambda$ are the same as their conventional counterparts; and $C$ is a function such that, for each edge $e \in E$, $C(e)$ is given by: “$= 0$”, “$= 100\%$”, “$\geq p\%$”, or “$\geq p$” ($p \geq 1$).

Remark that conventional pattern graphs are a special case where for each edge $e$, $C(e) \geq 1$ (only existential quantification). We omit $C(e)$ from each edge $e$ if it is an existential quantification.

It is clear to see that LR constraints are much closer to counting quantifiers with numeric aggregate (denoted shortly $CQs^+$). Hence, we conduct in the next a comparison between LR constraints and $CQs^+$ and we show how to extend our algorithm $TSim$ to handle pattern graphs with $CQs^+$. Since the other forms of $CQs$ are not too close to our problem, we consider in the next quantified pattern graphs with only numeric aggregates.

D.1 LR Constraints v.s. CQs$^+$

The limit of CQs$^+$ is that they specify the minimum number of children which must have all the same properties (child clone). Formally, given the edge $A \xrightarrow{\geq p} B$ of some pattern graph $Q$. This specifies that: 1) each data node $v$, that matches $A$, must have at least $p$ child nodes that match $B$; and 2) all these $p$ nodes must satisfy the same properties set that are defined over $B$ in $Q$. Moreover, $CQs^+$ are defined over children only.

An LR constraint, however, specifies the minimum number of children or parents that has some query node such that they have all the same label but not necessarily the same properties. In addition, any $CQ^+$ can be transformed to an LR constraint (Proposition [1]), but the inverse is not always possible as shows the next example.

Example 3. Consider the pattern graphs $Q_1$, $Q_2$, $Q_3$, and the data graph $G$ depicted in Fig. 5. The pattern graph $Q_1$ looks for each professor (Pr) which has supervised: 1) at least
two master students (MS) who have worked in the Cloud Computing (CC) area; and 2) at least two PhD students who had topics related to Collaborative Editing (CE) and Electronic Vote (EV) respectively. Remark that Q₁ is composed by one LR property and one CQ⁺. This CQ⁺ can be easily replaced by an LR constraint as follows: we replace the global child node q₂ by two copies of it, q₂ and q₂′, such that the properties defined in Q₁ over q₂ (child q₅ of type CC) must be duplicated over each copy of it. This transformation yields for the pattern graph Q₂. See that Q₁ and Q₂ are equivalent: by using the algorithm in [10], the matching of Q₁ over G returns the whole data graph G as match result, which is the same result returned by TSim(Q₂, G). However, it is clear that the LR constraint of Q₁ can not be replaced by the CQ⁺ “p ⩾ 2, phd” as done with the pattern graph Q₃. Thus, Q₁ and Q₃ are not equivalent: matching CQs via triple simulation that takes into account CPL relationships as well as CQs⁺. We show later that implementing this new definition requires just a simple extension of algorithm TSim.

Definition 5. Given a data graph G(V, E, λ) and a quantified pattern graph Q(Vₐ, Eₐ, λₐ, C) where C defines only CQs⁺. Then, G matches Q via triple simulation, denoted by Q ⩾ T G, if there exists a binary match relation S_T ⊆ V_Q × V s.t.:

1. For each (u, v) ∈ S_T, λₐ(u) = λ(v).
2. For each u ∈ V_Q there exists (u, v) ∈ S_T.
3. For each (u, v) ∈ S_T and for all simple edges (u, u₁), ..., (u, uₙ) ∈ Eₐ, there exists at least n distinct children v₁, ..., vₙ of v in G such that: (u₁, v₁), ..., (uₙ, vₙ) ∈ S_T.
4. For each (u, v) ∈ S_T and for all simple edges (u₁, u), ..., (uₙ, u) ∈ Eₐ, there exists at least n distinct parents v₁, ..., vₙ of v in G such that: (u₁, v₁), ..., (uₙ, vₙ) ∈ S_T.
5. For each (u, v) ∈ S_T and for each edge e = (u, u') in Eₐ with C(e) = “⩾ p”, there exists at least p distinct children v₁, ..., vₚ of v in G such that: (u', v₁), ..., (u', vₚ) ∈ S_T.

Intuitively, we enhance the old definition of triple simulation with the condition (5) in order to preserve CQs⁺ of Q. This condition requires that, for each edge u ⩾ p u' in Q, each match v of u in G must have at least p distinct children that match all the child u' of u.

D.2 Adapting TSim for CQs⁺

Given a quantified pattern graph Q(V, E, λ, C) where C represents only CQs⁺. A new definition of the procedure LR_Checking is given in Fig. in order to handle CQs⁺. Given a query node u with a potential match v. As explained above, we construct two bipartite graphs BG₁ and BG₂ that inspect the LR constraints defined over children and parents of u respectively. Recall that CQs⁺ are defined over children only. Thus, the equivalent LR constraint of each one is defined and included in BG₁ [lines 6-14]. For each child u' of u that is concerned by a CQ⁺ of cardinality p [line 6], we create p copies of u. Each subgraph in this match result satisfies CPL relationships as well as CQs⁺ of Q.
Procedure LR\_Checking($Q, G, u, v$)

**Input**: A QGP $Q(V_Q, E_Q, \lambda_Q, C)$ with only $CQ^+$, a data graph $G(V, E, \lambda)$, a node $u \in V_Q$ with a potential match $v \in V$.

**Output**: Whether LR constraints and $CQ^+$ defined over $u$ are satisfied by children and/or parents of $v$.

1: $BG_1 := (X_1 \cup Y_1, E_1)$; $BG_2 := (X_2 \cup Y_2, E_2)$; where $X_1 = Y_1 = X_2 = Y_2 = E_1 = E_2 = \emptyset$;
2: for each child $u'$ of $u$ with $u' \in LR(Q, u)$ do
3: $X_1 := X_1 \cup \{u'\}$;
4: Do $Y_1 := Y_1 \cup \{v'\}$; $E_1 := E_1 \cup \{(u', v')\}$; for each $(v', v) \in E$;
5: end for
6: /* CONSIDER THE $CQ^+$ DEFINED OVER THE CHILDREN OF $u$ */
7: $X_1 := X_1 \cup \{u_1, \ldots, u_p\}$; /*CREATE $p$ COPIES OF THE CHILD $u'$*/
8: for each $v' \in SIM(u')$ with $(v', v) \in E$ do
9: $Y_1 := Y_1 \cup \{v'\}$;
10: for each copy $u_i$ of $u'$ in $X_1$ do
11: $E_1 := E_1 \cup \{(u_i, v)\}$;
12: end for
13: end for
14: for each parent $u'$ of $u$ with $u' \in LR(Q, u)$ do
15: $X_2 := X_2 \cup \{u\}$;
16: Do $Y_2 := Y_2 \cup \{v\}$; $E_2 := E_2 \cup \{(u', v)\}$; for each $(v', v) \in E$;
18: end for
19: return true if ($CompleteMatch(BG_1)$ & $CompleteMatch(BG_2)$); and false otherwise.

**Figure 6** New version of procedure LR\_Checking to handle $CQ^+$.

In $X_1$ [line 7], each potential match of $u'$ is also a potential match of each copy of $u'$. For this reason, 1) we add into $Y_1$ each child $v'$ of $v$ that matches the child $u'$ of $u$; and 2) we create an edge between each copy $u_i$ and $v'$ to say that this copy can be matched by $v'$. The resulting bipartite graph $BG_1$ inspects: 1) the LR constraints defined over children of $u$; and 2) each LR constraint that results from the transformation of a $CQ^+$ defined over some child of $u$. If a complete matching exists over $BG_1$ then all these LR constraints are satisfied by children of $u$, i.e. all $CQ^+$ defined over children of $u$ are also satisfied (Proposition[1]).

**Example 4.** Consider the quantified pattern graph $Q$ and the data graph $G$ depicted in side (a) and (b) of Fig. 7 respectively. It is clear to see that the LR constraint, defined over the children $q_1$ and $q_2$ of $q_1$, is satisfied over $G$: by matching $q_1$, $q_3$, $q_4$ with $d_1$, $d_4$, and $d_5$ respectively. However, the $CQ^+$ “$q_1 \overset{2}{\rightarrow} q_2$” is not satisfied. The match $d_1$ of $q_1$ must have at least two child nodes such that: each one is labeled with $B$ and have a child node labeled with $C$. Consider the tuple $(q_1, d_1)$, the bipartite graph $BG_1$ constructed by the new procedure LR\_Checking is given in Fig. 7 (c). See that two copies of $q_2$ are created ($q_2$ and $q_2'$) and each one is connected to the unique match $d_3$ of $q_2$. Since there is no complete matching over $BG_1$, the procedure returns $false$ which means that the LR constraint and the $CQ^+$, that are defined over children of $q_1$, are not all satisfied by children of $d_1$.

The next result states that the problem of matching pattern graphs with numeric aggregates is in PTIME when it is treated as an extension of graph simulation, contrary to the NP-Completeness found in [10] when the problem is studied under subgraph isomorphism.
Figure 7 Satisfy CQs+ as LR constraints.

Theorem 5. Given a data graph $G(V, E)$ and a quantified pattern graph $Q(V_Q, E_Q, \lambda_Q, C)$ where $C$ defines only CQs+. Let $p$ be the largest cardinality of numeric aggregates $Q$. Algorithm TSim takes at most $O(|Q||G| + p, |V_Q|^3|V|^2 \sqrt{p, |V_Q| + |V|})$ time to decide whether $G \prec_T Q$ and to find the match result $M_T(Q, G)$.

Here algorithm TSim uses the new version of procedure LR_Checking given in Fig. 6. The overall time complexity of algorithm TSim, in case of pattern graphs with numeric aggregates, is bounded by $O(p^{1.5}|Q|^2|G|^{1.5})$ where $p$ is bounded as follows: $1 \leq p \leq |V|$. 