Process Similarity Algorithm Based on Transition Graph Edit Distance (Postprint)

Authors: Duan Rui, Fang Huan, Fang Xianwen, Zhan Yue

Date: 2019-01-28T00:00:00+00:00

Abstract

To improve the efficiency of querying and retrieving models from enterprise model repositories, we propose a process similarity algorithm based on transition graph edit distance. First, we present the concept of transition graph and its generation method. Second, we introduce the concept of edge length, where the costs of deleting and inserting edges are determined by the edge' s length; based on this, we define graph edit operations and their costs, and compute the minimum graph edit distance using a node matching algorithm. Then, we present the concept and calculation method for similarity between two process models. Finally, experiments verify the correctness of the algorithm, its satisfaction of seven similarity properties, and that the transition graph edit distance satisfies four distance properties.

Full Text

Preamble

Vol. 37 No. 4
Application Research of Computers

Process Similarity Algorithm Based on Editing Distance of Transition Graph

Duan Rui, Fang Huan, Fang Xianwen, Zhan Yue
(School of Mathematics & Big Data, Anhui University of Science & Technology, Huainan Anhui 232001, China)

Abstract: To improve the efficiency of querying and retrieving models from enterprise model libraries, this paper proposes a process similarity algorithm based on the edit distance of transition graphs. Firstly, the concept of transition graphs and their generation method are introduced. Secondly, the concept

of edge length is proposed, where the cost of deleting and inserting edges is determined by their length. Based on this, graph editing operations and their costs are defined, and a node matching algorithm is used to calculate the minimum graph edit distance. Then, the concept and calculation method for similarity between two process models are presented. Finally, experiments verify the correctness of the algorithm and its satisfaction of seven similarity properties, and confirm that the transition graph edit distance satisfies four distance properties.

Keywords: Petri nets; similarity measure; transition graph; graph edit distance

0 Introduction

As one of the three essential elements of an enterprise, business process model similarity measurement has always been a crucial research direction in workflow studies. For model repositories containing tens of thousands of models, efficient and accurate model retrieval methods become critical for business process management, which demands high-performance process similarity algorithms.

Existing process similarity algorithms primarily fall into three categories: (a) the most intuitive similarity measurement methods focusing on task labels, events, or other modeling elements; (b) approaches based on model topology, focusing on element sets and behavioral relationships between elements; and (c) algorithms based on behavioral semantics.

To evaluate algorithm performance, scholars and experts in process similarity have proposed seven fundamental similarity properties: sequential structure drift invariance, concurrent structure drift invariance, cyclic structure drift invariance, mutually exclusive structure drift invariance, span negative correlation, non-substitution-independent decreasing property, and cyclic sequence length negative correlation.

Zha et al. proposed a similarity algorithm based on transition adjacency relations called the TAR algorithm, which characterizes process model similarity by examining pairwise adjacency relations of process transitions. Yin et al. proposed an improved TAR algorithm called TAR++, which innovatively added importance coefficients to TAR, effectively satisfying some similarity properties that TAR could not meet. However, TAR++ uses a depth-first search method to assign importance to adjacent transition relations, resulting in factorial-level complexity. Moreover, TAR similarity has a relatively low upper bound.

Weidlich et al. proposed a similarity measurement method based on behavioral profiles (BP). This method defines weak ordering and proposes a series of relations based on weak order relations, collectively called behavioral profiles. This approach extends adjacent transition relations using behavioral profiles, effectively satisfying some similarity properties that TAR cannot meet, but still fails to satisfy all properties. The SSDT method constructs a shortest follow-up distance matrix for tasks to measure process similarity. However, each time

SSDT calculates process similarity, it needs to perform corresponding expansions based on matrix rank to ensure equal ranks between two matrices, making its performance less than optimal.

The CF algorithm can satisfy all similarity properties, but its time complexity is high. Graph matching and graph edit distance have always been important means for model similarity measurement. Dijkman et al. described four graph matching algorithms: greedy algorithm, exhaustive algorithm with pruning, process heuristic algorithm, and A* algorithm. Experimental evaluation shows that the greedy algorithm consumes far less time than other algorithms, the A* algorithm has the highest accuracy, and the exhaustive algorithm with pruning consumes far more time than others. Graph edit distance refers to the operation cost required to transform one graph into another. Calculating process similarity requires the minimum graph edit distance, and graph matching algorithms find this minimum by establishing one-to-one correspondences between nodes during model comparison.

Addressing existing problems in similarity measurement algorithms—such as failure to satisfy similarity properties, high time complexity, state space explosion, low flexibility, and discrepancies between calculated and expected values—this paper proposes a novel process similarity algorithm based on transition graph edit distance (TGED). The main contributions are: (a) transforming Petri net models into transition graphs through place mapping, where places in Petri net models are mapped into edges in transition graphs; (b) introducing for the first time the concept of edge length, where the cost of editing edges differs based on their length; (c) taking the minimum operation cost required to transform one transition graph into another as the edit distance, and calculating similarity based on this; (d) the TGE D algorithm has a larger upper bound and lower time complexity.

---

1 Preliminary Knowledge

This paper uses Petri nets as a formal modeling and analysis tool. Before introducing the TGE D algorithm, we first present preliminary knowledge about Petri nets.

Definition 1 (Labeled Petri Net). A five-tuple (L, N, P, T, F, Σ) is called a labeled Petri net, where: P is the set of places; T is the set of transitions; $F \subseteq (P \times T) \cup (T \times P)$ is the flow relation; Σ is the set of transition labels; $\ell : T \to \Sigma \cup \{\epsilon\}$ is the label function. For $x \in P \cup T$, the preset of x is denoted as $\bullet x = \{y \mid (y, x) \in F\}$, and the postset of x is denoted as $x\bullet = \{y \mid (x, y) \in F\}$. An element $x \in P \cup T$ is called a node of the Petri net, and $f \in F$ is called an edge. Edges adjacent to x are called edges of x, including input and output edges. Adding an initial marking M_0 to a labeled Petri net yields $(L, N, P, T, F, \Sigma, M_0)$, called a labeled Petri net system, where $M_0 : P \to \mathbb{N}^+$ is a non-negative integer set.

Definition 2 (Workflow Net). A triple $WF = (N, N_i, N_o)$ is called a workflow net, where: N is a labeled Petri net system; N_i is the start place; N_o is the end place. (a) N_i and N_o are unique; (b) for all $x \in P \cup T$, there exists a path from N_i to N_o containing x.

All models discussed in this paper are based on safe workflow nets of Petri nets. A Petri net is called safe if all places are bounded with a bound of 1, meaning a place contains at most one token. The initial marking serves as the triggering condition for the entire system model—under the initial marking, the start place contains one token while other places contain none. To facilitate similarity calculation, we propose a graph consisting only of transitions and edges without places, called a transition graph.

Definition 3 (Graph Edit Distance). Given two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, the minimum cost required to transform G_1 into G_2 is called the edit distance between G_1 and G_2, denoted as $edist(G_1, G_2)$. To determine graph edit distance, reasonable graph edit operations and their costs must be defined. This paper summarizes existing graph edit operations and introduces a new concept: edge length, where the cost of deleting and inserting edges is determined by their length.

Definition 4 (Graph Edit Operations). Given a graph $G = (V, E)$, the basic operations for editing graph G are defined as: node deletion, insertion, substitution, and edge deletion, insertion. $\epsilon(u)$ denotes deleting node u, $\epsilon(v)$ denotes inserting node v, and $u \to v$ denotes substituting node u with node v. Similarly, edge deletion and insertion are denoted. As shown in [Figure 1: see original paper], transforming from Figure 1(a) to (b) involves deleting edge (a, f), from (b) to (c) substituting node g with node c, and from (c) to (d) inserting edge (d, e). After a series of basic graph edit operations, Figure 1(a) is transformed into (d).

---

2 Similarity Calculation Based on Transition Graphs

To calculate the similarity between two process models, we first need to generate transition graphs from their Petri net models, then define reasonable basic graph edit operations and their costs for transition graphs, and finally obtain the minimum cost as the graph edit distance.

2.1 Transition Graph Generation

Definition 5 (Transition Graph). Given a workflow net $WF = (N, N_i, N_o)$, where $N = (P, T, F, \Sigma, M_0)$ is a labeled Petri net system; $G = (V, E)$ is a transition graph, where V is the node set and $E \subseteq V \times V$ is the edge set. The generation method is:

For each place $p \in P$, map it to an edge $(p_\bullet, \bullet p)$, where the front-end node is $p_\bullet$ and the back-end node is $\bullet p$. Specifically, if $|\bullet p| > 1$, the front-end node

of the corresponding edge is determined by the back-end node of the preceding edge, or serves as the end of a mutually exclusive structure, corresponding to j edges, or triggers a cyclic structure; if $|p \bullet| > 1$, the back-end node of the corresponding edge will lead to j edges, each controlled by a transition in that node, called a control transition.

To handle cases where workflow nets begin or end with special structures, we manually add a start place p_s and transition t_s, and an end place p_e and transition t_e, along with four edges: (t_s, p_s), (p_s, t_s), (t_e, p_e), and (p_e, t_e).

As shown in [Figure 2: see original paper], the left side displays four typical Petri net model structures: sequential, mutually exclusive, cyclic, and concurrent, with their corresponding transition graphs on the right. In Figure 2(a), the left side shows a sequential structure. After manual addition of start place p_s and transition t_s, and end place p_e and transition t_e, along with four edges (t_s, p_s), (p_s, t_s), (t_e, p_e), and (p_e, t_e), the new model has 4 places besides p_s and p_e. According to Definition 5, this corresponds to 4 edges, resulting in the transition graph shown on the right of Figure 2(a). Figure 2(b) shows a mutually exclusive structure embedded with a sequential structure. After manual transformation: for place p_1, $|\bullet p_1| = 1$, and for place p_2, $|p_2 \bullet| > 1$, the back-end node of the edge corresponding to place p_2 will lead to two edges controlled by t_3 and t_4 respectively, as shown on the right of Figure 2(b). Figure 2(c) shows a cyclic structure. For place p_1, $|\bullet p_1| > 1$, then the front-end node of the edge corresponding to place p_1 is determined by the back-end node of the edge corresponding to place p_2, i.e., $\bullet p_1 = p_2 \bullet$, where p_1 triggers a cyclic structure. Figure 2(d) shows a concurrent structure. In the transition graph definition provided in this paper, sequential and concurrent structures are the easiest to handle, with the right side of Figure 2(d) showing its corresponding transition graph.

2.2 Behavioral Semantics of Transition Graphs

As a place-free simplification of Petri net models, each node in a transition graph may contain multiple transitions because a place can trigger multiple mutually exclusive branches. Similarly, a node in a transition graph may have multiple edges: (a) serving as the end of a mutually exclusive structure embedded with other structures; (b) where the triggered mutually exclusive branches have cyclic structures; (c) concurrent structures. This paper calls a complete execution sequence from the manually added transition t_s to t_e a statement, and all statements of a transition graph constitute the language of the transition graph. The behavioral preservation interpretation of transition graphs is as follows:

(a) All occurrence sequences of a transition graph constitute its behavioral semantics. Edges in transition graphs are controlled by transitions, and transitions on edges determine that the edge can only be triggered by the control transition. As shown in Figure 2(b), the occurrence sequence of the transition graph is: $\langle \{t_s\}, \{t_1\}, \{t_2\}, \{t_3\}, \{t_e\} \rangle$. If an edge is empty, since

the node contains only one transition, after that transition is triggered, all edges leading from it can be triggered. As shown in Figure 2(d), the occurrence sequence of the transition graph is: $\langle\{t_s\},\{t_1\},\{t_4\},\{t_2,t_3\},\{t_e\}\rangle$.

(b) Node triggering conditions in transition graphs: edges with a node as their back-end node must be triggered first. Transition graphs containing only one transition have multiple edges leading out concurrently, which can be explained by the previous point—edges leading from such nodes are by default controlled by the node element.

2.3 Graph Edit Operations and Their Costs

As mentioned earlier, the basic graph edit operations provided in this paper are: node deletion, insertion, substitution, and edge deletion, insertion. This section assigns reasonable costs to these operations. Node deletion and insertion are considered unit basic operations, including control transition insertion, deletion, and substitution, with a cost of 1. The following details node substitution and edge deletion/insertion.

2.3.1 Node Substitution Node substitution involves the similarity of transition labels in nodes. The cost of deleting, inserting, or substituting a unit character in a label is 1. Transforming one transition label into another through a series of string operations, the minimum cost required is the edit distance between the two transition labels. The ratio of this edit distance to the maximum label length is the node substitution operation cost. If two transition labels are completely different, the substitution operation cost is 1.

As shown in Figure 3, transforming the transition label "Lookingforfood" from Figure 3(a) to "Findfood" in Figure 3(b) requires a minimum operation cost of 10, i.e., substituting 4 unit characters and deleting 6 unit characters. The entire node substitution cost is $\frac{10}{14} = 0.714$.

2.3.2 Edge Deletion and Insertion This paper first proposes the concept of edge length to describe the cost of deleting and inserting an edge. The length of an edge corresponding to place p is 1. A longest simple path between manual transitions t_s and t_e is selected as the backbone, with all edges on the backbone having length 1. The length of a branch crossing the backbone is its approximate length, and node length is the sum of edge lengths.

As shown in Figure 3, transforming from Figure 3(a) to (b) requires substituting node a and deleting node b and two edges. When nodes are infinitely close, the length of these two edges is approximately 1, so the deletion cost is 1.

2.4 Similarity Calculation

From Definition 3, the minimum cost to transform one graph into another is their edit distance. The core of the similarity measurement method based on

transition graph edit distance is to find the minimum transition graph edit distance.

Definition 6 (Similarity). Given two workflow nets W_1 and W_2, their corresponding transition graphs are $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$. Let $|G|$ denote the modulus of graph G, where $|G| = |G.V| + |G.E| + |G.CT|$, with $|G.E|$ representing the sum of approximate lengths of all edges in graph G, and $|G.CT|$ representing the number of control transitions in graph G. Then the similarity is:

$$sim(G_1, G_2) = 1 - \frac{edist(G_1, G_2) - 2}{\max(|G_1|, |G_2|)}$$

In the denominator of the similarity calculation formula in Definition 6, subtracting 2 eliminates the influence of manually added transitions. As shown in Figure 3, $edist(a, b) = 3.714$, so $sim(a, b) = 1 - \frac{3.714-2}{\max(16,16)} = 0.735$.

---

3 TGE D Algorithm

3.1 Algorithm Design

This section presents the TGE D algorithm for process similarity based on transition graph edit distance. From the previous discussion, the algorithm' s core is generating transition graphs and calculating their edit distance.

The algorithm first manually transforms workflow nets (line 1). Lines 2-16 generate transition graphs for the transformed workflow nets, where lines 2-4 create and initialize the transition graph, lines 5-15 calculate the transition graph of the workflow net according to Definition 5, line 16 calculates the transition graph of the other workflow net, line 17 calculates the minimum edit distance between the two transition graphs, and lines 18-19 calculate similarity according to Definition 6.

Calculating the minimum operation cost to transform two transition graphs (line 17) is not a simple process. Literature [6] proposes node matching based on the A* search algorithm. During iteration, it continuously uses the currently generated partial mapping for node matching, selects the one with minimum cost (maximum similarity) for expansion, and finally obtains the optimal solution. This algorithm defines a lower bound estimate of the edit distance for remaining nodes for "pruning." Literature [9] provides four matching algorithms to solve "mapping that induces maximum similarity," i.e., minimum graph edit distance. Based on the average accuracy and running time of the four algorithms, this paper adopts the greedy algorithm and A* algorithm.

3.2 Algorithm Time Complexity Analysis

The workflow net transformation method manually adds two transitions, two places, and four edges, with constant time complexity. Creating and initializing transition graphs also has constant time complexity. Lines 5-16 of the algorithm generate transition graphs with time complexity $O(|P|+|T|+|F|)$, which can be obtained through amortized analysis. The greedy algorithm for calculating minimum graph edit distance has time complexity $O(|V_1| \times |V_2|)$. The A* algorithm for minimum graph edit distance has worst-case time complexity $O(|V_1|! \times |V_2|!)$, but in practice, due to the use of the valuation function, the A* algorithm' s running time is far lower than the worst-case time complexity, approaching the optimal time complexity.

---

4 Experiments and Evaluation

The experimental models in this paper mainly come from the SAP model repository, from which 230 models were randomly selected as experimental objects. To verify the satisfaction of similarity properties, another 70 models were manually compiled, resulting in a total of 300 process models as experimental data.

4.1 Experimental Design

The experimental machine environment is: Intel(R) Core(TM) i5-7300HQ CPU @ 2.50 GHz, 8.00 GB RAM, 64-bit operating system. Using the open business process model management framework BeehiveZ system as the tool, which has various process management functions, this paper mainly uses its query function (i.e., retrieval function). The experimental data is divided into two categories: the first category consists of 230 models randomly obtained from the SAP model repository to verify the feasibility of TGE D; the second category consists of 70 manually compiled models to verify the satisfaction of similarity properties by the TGE D algorithm.

First, experiments verify the correctness of the TGE D algorithm, i.e., whether it can output a number between [0,1] when given two workflow nets. Second, graph edit distance properties need to be verified, focusing on the triangle inequality property. Finally, the satisfaction of similarity properties by the TGE D algorithm is verified.

4.2 Distance Property Verification

The graph edit distance proposed in this paper is verified against distance properties. Similarity measures should satisfy four distance properties: symmetry, reflexivity, non-negativity, and triangle inequality.

Symmetry: The similarity between two workflow nets is unique, i.e., $\forall W_1, W_2, edist(W_1, W_2) = edist(W_2, W_1)$. All edit operations are symmetric,

so the minimum edit operation cost between two models is symmetric, and their TGE D similarity satisfies symmetry.

Reflexivity: The similarity between a workflow net and itself is 1, i.e., $\forall W, edist(W,W) = 0 \Rightarrow sim(W,W) = 1$. Two identical graphs can be transformed into each other without any operations.

Non-negativity: The similarity between two graphs is calculated as: $sim(G_1,G_2) = 1 - \frac{edist(G_1,G_2)-2}{\max(|G_1|,|G_2|)}$. Since $edist(G_1,G_2) \leq \max(|G_1|,|G_2|) - 2$ always holds, $sim(G_1,G_2) \geq 0$.

Triangle Inequality: $\forall W_1, W_2, W_3, edist(W_1,W_2) \leq edist(W_1,W_3) + edist(W_3,W_2)$. Given any three models and their three distances, the sum of any two distances is greater than or equal to the third distance. This is the most important distance property for similarity and requires experimental verification.

First, we define the triangle inequality satisfaction rate. Assuming there are n models in the experimental dataset, taking any 3 models from these n models yields C_n^3 combinations. If n' model combinations satisfy the triangle inequality, the triangle inequality satisfaction rate is n'/C_n^3. To better evaluate the triangle inequality satisfaction rates of different algorithms, this paper conducts three experiments to obtain three groups of triangle inequality satisfaction rates. Each time, a certain number of models are randomly selected from the model dataset as the experimental dataset. The numbers of models selected in the three experiments are 94, 117, and 123, respectively, denoted as Dataset 1, 2, and 3. To better demonstrate the advantages of the TGE D algorithm, in addition to the similarity algorithms mentioned in the introduction, this section adds two more algorithms as experimental objects: the causal footprint algorithm (CF) and the complete firing sequence algorithm (CFS).

As mentioned earlier, the cost of inserting and deleting edges is determined by their length. To accurately verify the triangle inequality satisfaction rate, approximate values cannot be used when calculating graph edit distance. This paper uses $n+1$ to represent a length slightly larger than n, i.e., the cost of inserting and deleting them. The TGE D algorithm and other algorithms are used for triangle inequality satisfaction rate experiments, with results shown in [Figure 4: see original paper]. In terms of triangle inequality satisfaction rate, CF performs less well than other algorithms, and the TAR++ algorithm does not achieve 100% satisfaction rate on Dataset 1. The best performance comes from TGE D(G) (TGE D using greedy algorithm for node matching) and TGE D(A) (TGE D using A* algorithm for node matching).

A more comprehensive algorithm performance comparison also includes algorithm running time, as shown in [Figure 5: see original paper]. The TGE D algorithm is divided into TGE D(G) using greedy algorithm for node matching and TGE D(A) using A* algorithm. TGE D(G) requires significantly less time than other algorithms, while TGE D(A) requires slightly more time than oth-

ers but offers higher accuracy. Combining triangle inequality satisfaction rates, we conclude that the TGE D algorithm slightly outperforms other algorithms in overall performance. In triangle inequality satisfaction rate, the greedy algorithm is slightly worse than exhaustive and A* algorithms but has certain advantages in running time. Therefore, when high accuracy is required, the A* algorithm is chosen for node matching; when accuracy requirements are not high, the greedy algorithm is chosen to save time.

4.3 Similarity Property Verification

Currently proposed similarity properties mainly include: sequential structure drift invariance, concurrent structure drift invariance, mutually exclusive structure drift invariance, cyclic structure drift invariance, span negative correlation, non-substitution-independent decreasing property, and cyclic sequence length negative correlation. The experimental model dataset consists of 70 manually compiled process models.

Property 1 (Sequential Structure Drift Invariance). No matter where in the sequential structure a new transition is inserted to obtain a new sequential structure model, the similarity between the new model and the original model remains equal.

A sequential structure model containing 10 transitions is manually created as the original model N, as shown in [Figure 6: see original paper] (for brevity, some transitions are omitted). New transitions are inserted between transitions of the original model to obtain 11 new models (figures omitted). Converting all models to transition graphs, the edit distance from each new model' s transition graph to the original model' s transition graph is 2, meaning the similarity between all new models and the original model is 0.913.

Property 2 (Concurrent Structure Drift Invariance). When the sequential structure in a process model is modified, the similarity between the new model and the original model remains equal regardless of which part of the sequential structure the concurrent structure branch is added to.

Property 3 (Mutually Exclusive Structure Drift Invariance). When the sequential structure in a process model is modified, the similarity between the new model and the original model remains equal regardless of which part of the sequential structure the mutually exclusive structure branch is added to.

Property 4 (Cyclic Structure Drift Invariance). When the sequential structure in a process model is modified, the similarity between the new model and the original model remains equal regardless of which part of the sequential structure the cyclic structure branch is added to.

In fact, the first four properties can be summarized as one property: structure drift invariance. However, some algorithms can only satisfy part of these four properties, so they are described separately for distinction. Models N_1, N_2, and N_3 in [Figure 7: see original paper] are new models obtained by modifying

the original model according to Properties 2, 3, and 4, respectively, with one example shown in Figure 7 and others omitted. For Property 2, regardless of where the concurrent structure branch is added, the corresponding transition graph edit distance is 3. For Property 3, regardless of where the mutually exclusive structure branch is added, the corresponding transition graph edit distance is 0.5. For Property 4, regardless of where the cyclic structure branch is added, the corresponding transition graph edit distance is 3.5. Operations with cost 0.5 come from node substitution, i.e., the cost of inserting another transition into a node containing one transition.

Property 5 (Span Negative Correlation). When adding a mutually exclusive structure branch to the sequential structure of a model, the larger the span of this branch in the original model structure, the smaller the similarity between the new model and the original model.

As shown in [Figure 8: see original paper], adding a mutually exclusive structure with span 2 to N yields model N_1, with $sim(N, N_1) = 0.820$. From the previous property, we know $sim(N, N_2) = 0.976$. Increasing the span of the mutually exclusive structure yields another series of models, verifying the correctness of span negative correlation.

Property 6 (Non-Substitution-Independent Decreasing Property). When adding mutually exclusive structure branches to the sequential structure of a model, the more branches added, the smaller the similarity between the new model and the original model.

As shown in [Figure 9: see original paper], model N_4 adds two mutually exclusive branches to the same part of N, and N_5 adds another mutually exclusive branch to the mutually exclusive structure part of N_4, with $sim(N, N_4) < sim(N, N_5)$.

Property 7 (Cyclic Sequence Length Negative Correlation). When modifying the sequential structure of a process model by adding a cyclic structure, the larger the span of the cyclic structure, the smaller the similarity between the new model and the original model.

Property 7 is similar to Property 5, and experimental verification shows that the TGE D algorithm satisfies all seven similarity properties.

shows the satisfaction of the seven similarity properties by each algorithm, where $\surd$ indicates satisfaction and $\times$ indicates non-satisfaction. As shown in Table 1, the SSDT algorithm needs to perform corresponding expansions based on matrix rank each time it calculates process similarity to ensure equal ranks between two matrices. The CFS algorithm enumerates all possible execution sequences caused by concurrent tasks. Therefore, the TGE D algorithm (the algorithm in this paper) satisfies all similarity properties, while TGE D(A) has slightly longer running time than TGE D(G). The TAR++ algorithm uses depth-first search to assign importance to adjacent transition relations, with worst-case time complexity of $O(n!)$, which often occurs in practice. Combining each algorithm' s triangle inequality satisfaction rate and running time, we further conclude that

the TGE D algorithm slightly outperforms other algorithms.

5 Conclusion

To improve the efficiency of querying and retrieving models from enterprise model libraries and address existing problems in process similarity algorithms, this paper proposes a process similarity algorithm based on transition graph edit distance. The algorithm transforms models into simple transition graphs, calculates the minimum transition graph edit distance, and derives similarity. Additionally, this paper introduces for the first time the concept of edge length, where the cost of deleting and inserting edges is determined by their length, defines graph edit operation costs based on this, and uses greedy and A* algorithms to calculate minimum graph edit distance. Experiments demonstrate that the proposed algorithm slightly outperforms other algorithms in overall performance.

Future work aims to deeply investigate various similarity measurement methods, extend similarity properties, optimize the TGE D algorithm or propose new and better algorithms to make similarity more aligned with domain expert evaluations.

References

[1] Lu Yahui, Yu Haofei, Ming Zhong, et al. A similarity measurement based on structure of business process [C]// Proc of IEEE International Conference on Computer Supported Cooperative Work in Design. 2016.

[2] Montani S, Leonardi G, Quaglini S, et al. A knowledge-intensive approach to process similarity calculation [J]. Expert Systems with Applications, 2015, 42 (9): 4207-4215.

[3] Wang Jianmin, Jin Tao, Wong R K, et al. Querying business process model repositories [J]. World Wide Web-internet & Web Information Systems, 2014, 17 (3): 427-454.

[4] Wang Shuhao, Wen Lijie, Wei Daiseng, et al. SSDT matrix-based behavior similarity algorithm for process model [J]. Computer integrated manufacturing system, 2013, 19 (8): 1822-1831.

[5] Yin Ming, Wen Lijie, Wang Jianmin, et al. Process similarity algorithm based on importance of transition adjacent relations [J]. Computer integrated manufacturing system, 2015, 21 (2): 344-358.

[6] Wang Zixuan, Wen Lijie, Wang Shuhao, et al. Similarity measurement for process models based on transition-labeled graph edit distance [J]. Computer integrated manufacturing system, 2016, 22 (2): 343-352.

[7] Zha Haiping, Wang Jianmin, Wen Lijie, et al. A workflow net similarity measure based on transition adjacency relations [J]. Computers in Industry, 2010, 61 (5): 463-471.

[8] Weidlich M, Elliger F, Weske M. Generalised computation of behavioural profiles based on Petri-net unfoldings [M]// Web Services and Formal Methods. Berlin: Springer, 2010: 101-115.

[9] Dijkman R, Dumas M. Graph matching algorithms for business process model similarity search [C]// Proc of International Conference on Business Process Management. [S.l.]:Springer-Verlag, 2009: 48-63.

[10] Dijkman R, Dumas M, Dongen B V, et al. Similarity of business process models: metrics and evaluation [J]. Information Systems, 2011, 36 (2).

[11] Jin Tao, Wang Jianmin, Wen Lijie. Efficiently querying business process models with beehiveZ [C]// Proc of Demo Track of the 9th Conference on Business Process Management. 2011.

[12] Wu Nianhua, Jin Tao, Cha Haiping, et al. BeehiveZ: an open business process model management framework. [J]. Computer Research and Development, 2010, 47 (z1): 450-454.

[13] Dong Zihe, Wen Llijie, Huang Haowei, et al. CFS: a behavioral similarity algorithm for process models based on complete firing sequences [J]. Journal of Software, 2015: 202-219.

Note: Figure translations are in progress. See original paper for figures.