TAGER: Transition-labeled Graph Edit Distance Similarity Measure on Process Models

Zixuan Wang*, Lijie Wen*, Jianmin Wang*, and Shuhao Wang*

*School of Software, Tsinghua University, Beijing 100084, P.R. China
iamwangzixuan@hotmail.com,wenlj00@mails.tsinghua.edu.cn,
jimwang@tsinghua.edu.cn,shudiwsh2009@gmail.com

Abstract. Although several approaches have been proposed to compute the similarity between process models, they have various limitations. We propose an approach named TAGER (**T**ransition-l**A**beled **G**raph **E**dit distance similarity Measu**R**e) to compute the similarity based on the edit distance between coverability graphs. As the coverability graph represents the behavior of a Petri net well, TAGER, based on it, has a high precise computation. Besides, the T-labeled graphs (an isomorphic graph of the coverability graph) of models are independent, so TAGER can be used as the index for searching process models in a repository. We evaluate TAGER from efficiency and quality perspectives. The results show that TAGER meets all the seven criteria and the distance metric requirement that a good similarity algorithm should have. TAGER also balances the efficiency and precision well.

Keywords: Business Process Model, Transition-labeled Graph, Edit Distance, Behavioral Similarity

1 Introduction

Repositories with hundreds and even thousands of process models become increasingly common [**?**] and the organizations reach higher levels of Business Process Management (BPM) maturity [**?**]. There is an urgent requirement for searching the specific models among the process repository. Although a variety of techniques that can manage these repositories have been proposed already, the requirement for precise similarity measure is becoming more and more urgent. There are several approaches to measure the similarity between process models. A general classification for those algorithms [**?**] is as follows, according to which kind of information these algorithms are mainly based on.

- Text similarity: based on a comparison of the labels that appear in process models (e.g., task labels, event labels), using either syntactic or semantic similarity metrics, or a combination of them.
- Structural similarity: based on the topology of process models, possibly taking text similarity into account as well.
- Behavioral similarity: based on the execution semantics of process models.

Because the similarity measures based on text or structure only consider the label set or topological structure, they lack lots of information. Comparatively, the similarity measures based on behavioral features can have a better performance and provide a more convincing result. The existing behavioral similarity algorithms have various advantages. However, they have limitations, too. The PTS similarity measure algorithm [?], proposed by Jianmin Wang in 2010, is based on labeled Petri net. It defines three types of principal transition sequences and computes a result by weighting the similarity value of each type. But this approach considers the loop structure and other structures separately. As a result, it breaks the behavior semantics and it cannot handle Petri nets with loop structures effectively. TAR similarity algorithm [?] represents a model's behavior by transition adjacency relations. It computes the similarity by Jaccard coefficient of two TAR sets but cannot tackle the long-distance dependences between transitions. BP algorithm [?] gives the similarity based on behavioral profiles between transitions. However, it cannot distinguish the loop and parallel structures. Furthermore, it cannot deal with process models with invisible tasks.

TAGER, as a behavioral similarity measure, chooses Petri net as its modeling notation. The Petri net analysis tool, coverability graph [?], can represent the behavioral features of a Petri net well. However, the transition labels as the most useful information are stored on a coverability graph's edges. So we define the Transition-labeled graph (T-labeled graph for short), the isomorphic graph of a coverability graph, to emphasize the transition label on the edges of a coverability graph. Because graph edit distance is a classical method to measure the difference between two graphs, so TAGER choose to measures the similarity between T-labeled graphs by using their graph edit distance. And TAGER modifies and extends the traditional edit operations to get a precise similarity. The graph edit distance generally has been used in structural similarity measure, however, we use it based on the coverability graph which is the behavioral feature of the Petri net. Besides, compared to the traditional edit operations, we redefine substitute operation and consider some common behavior patterns. Our main contributions in this paper are as follows:

1. We propose an efficient and effective approach named TAGER to compute the behavioral similarity of process models. It is based on the edit distance between T-labeled graphs, which contain all behavior information of Petri nets.
2. To improve the precision of similarity measure, we redefine the graph edit operations [?] according to process models' structural features. We take the type and scale of vertices into account when redesigning the vertex substitution operation.
3. We define the similarity measure by combining the model scale with graph edit distance and give a sound computation approach with the best weight arrangement obtained by experiments.

The remainder of the paper is structured as follows. Section 2 formulates the problem and introduces the relevant concepts which will be used throughout the paper. Section 3 presents our algorithm which is used to measure the similarity. Section 4 presents the experimental evaluation of our algorithm. Section 5 concludes this paper and sketches some future work.

2 Preliminaries

This section presents the notions of process models used throughout this paper.

2.1 Petri net

Petri net was originally introduced by Carl Adam Petri [?] as a tool for modeling basic structures like sequence, conflict, parallel and loop which can be used to study the parallel, distributed and random system [1]. A Petri net is consisted of places and transitions. For each node, a unique identifier (like P_1 or T_2 in Figure 1) is associated. For the remainder of this paper, we assume that the set of possible identifiers is denoted as U.

Definition 1. ***(Petri net)*** *Let $P, T \subseteq U$ be finite and disjoint sets such that $P \cup T \neq \varnothing$ and $P \cap T = \varnothing$, let $F \subseteq (P \times T) \cup (T \times P)$. The tuple $N = (P, T, F, M_0)$ is a system, where P is the set of places, T is the set of transitions, F is the set of arcs and M_0 as a mapping from P to $\mathbb{N}$ is the initial marking of the net. $\mathbb{N}$ is the set of non-negative integers, i.e., $\{0, 1, 2, ...\}$.*

According to Definition 1, we introduce labeled Petri net. For each transition, a unique label is associated which indicates the action name.

Definition 2. ***(Labeled Petri net)*** *Let $N = (P, T, F, M_0)$ be a Petri net, the tuple $N = (P, T, F, M_0, A, l)$ is a labeled Petri net, where A is a finite set of action names, function l is a mapping from T to $A \cup \{\varepsilon\}$, ε represents an empty label of a dumb or silent transition.*

Figure 1 gives two samples of labeled Petri nets, i.e., M_0 and M_4. For example, P_0, P_1, P_2 and P_3 are the places of M_4 and T_0, T_1, T_2 and T_3 are the transitions of M_4.

Definition 3. ***(Workflow net)*** *A Petri net $N = (P, T, F, M_0)$ that models the control-flow dimension of a workflow is called a workflow net (WF-net for short), which should satisfy the following conditions:*

- *There is one unique source place i such that $\bullet i = \varnothing$;*
- *There is one unique sink place o such that $o \bullet = \varnothing$;*
- *Every node $x \in P \cup T$ is on a path from i to o.*

Fig. 1. Two sample labeled Petri nets

2.2 Coverability graph

Coverability graph is an effective model to represent a Petri net's behavioral features. The markings of the Petri net are mapping to the vertices of the coverability graph and the transitions of the Petri net are mapped to the edges of the coverability graph. A coverability graph differes from the topological structure of a Petri net by having the behavioral features. We can compute a coverability graph by the following steps [**?**].

Definition 4. ***(Coverability graph)*** *Let $N = (P, T, F, M_0, A, l)$ be a labeled Petri net, let $S = (N, I, O)$ be a net system, I is the system's initial state, whereas the marking O is its successful terminal state. Let $V \subseteq \mathbb{M}$ where $\mathbb{M}$ is the marking set of N, let $E \subseteq (V \times T \times V)$ be a set of $Transition-labeled$ arcs, and let $G = (V, E)$ be a graph which can be constructed as follows:*

1. *Initially, $V = \{I\}$ and $E = \varnothing$.*
2. *Take an extended marking (the new marking after firing any enabled transition at current marking) $M \in V$ and a $t \in T$ such that $M[t\rangle$ and no extended marking M_1 exists with $(M, t, M_1) \in E$. Let $M_2 = M_1 - \bullet t + t\bullet$. Add M_3 to V and (M, t, M_3) to E, where for every $p \in P$:*
   (a) $M_3(p) = \omega$, if there is a node $M_1 \in V$ such that $M_1 \leq M_2$, $M_1(p) < M_2(p)$, and there is a directed path from M_1 to M in G;
   (b) $M_3(p) = M_2(p)$, otherwise.

Repeat this step until no new arcs can be added.

Graph G is called a coverability graph of a net system S.

For example, we can capture the coverability graphs of M_0 and M_4 in Figure 1, which are shown in Figure 2.

(a) M_0

(b) M_4

Fig. 2. Corresponding coverability graphs

(a) M_0

(b) M_4

Fig. 3. Corresponding T-labeled graphs

3 The new algorithm TAGER

Our algorithm named TAGER is based on coverability graph which can analyze a Petri net without losing structural information. As the graph edit distance is a classical approach to measure the difference between two graphs, we choose it as the crucial factor on measuring two process models' similarity. In this section, we give an overview of TAGER and introduce the details of the crucial computing steps. We analyze the time complexity of TAGER and discuss its limitations in the end of this section.

3.1 The algorithm overview

To compute the graph similarity of two process models, we need to find a mapping that gives a maximal similarity match and a minimal graph edit distance. The TAGER algorithm computes the similarity between two process models expressed as Petri nets in the following steps:

1. Building the coverability graph for each Petri net;

2. Converting the coverability graphs to T-labeled graphs, which will be defined in the following subsection.
3. Using the modified greedy algorithm to compute the graph edit distance between the two T-labeled graphs.
4. Computing the similarity with the weights of the edit distance and the scale of graphs.

We can finish step 1 and step 2 according to Definition 4 in Section 2 and Definition 5 in Section 3.2 respectively. Step 3 can be computed on the foundation we give in Section 3.3 and via the method we will introduce in Section 3.4. Step 4 will be shown in Section 3.5.

3.2 The T-labeled graph

As TAGER is defined as a behavioral measure, we need the transition information which are stored on the edges in the coverability graph. Using the coverability graph directly will make the computation of the edit distance complex, so we transform the coverability graph to its isomorphic graph, which we named *T-labeled graph.* We just move the labels of transitions to their successive vertices.

Definition 5. ***(T-labeled graph)*** *Let $N = (P, T, F, M_0, A, l)$ be a labeled Petri net and $G_{coverability} = (V, E)$ be its coverability graph, where V stores the current markings and E stores the transitions' labels. Let $\mathcal{L}$ be the set of transition labels. We define T-labeled graph $G = (V, E, \lambda)$ as follows:*

- *V is the vertex set of $G_{coverability}$;*
- *E is the edge set of $G_{coverability}$;*
- *$\lambda : V \to 2^{\mathcal{L}}$ is a function that maps vertices to label sets.*

Now all transitions' labels are stored in vertices and the edges are just be used as the connectors of vertices. Let $G = (V, E)$ be a graph and $n \in V$ be a node: $\bullet n = \{m | (m, n) \in E\}$ is the pre-set of n, while $n\bullet = \{m | (n, m) \in E\}$ is the post-set of n. Source node is a node with an empty pre-set and sink node is a node with an empty post-set. According to Definition 3, a T-labeled graph can only has one source node and one sink node. We can convert coverability graphs in Figure 2 to their T-labeled graphs, which are shown in Figure 3.

3.3 The graph edit operations

The graph edit distance is computed on the foundation of edit operations, in which we redefine in TAGER algorithm. Besides, we give the equation for computing the edit distance between two graphs.

The edit operations The edit operations which have been mentioned in this paper include vertex insertion/deletion, vertex substitution and edge insertion/deletion. The cost of vertex insertion/deletion is assigned one in TAGER. Furthermore, the traditional computation of the cost in the other two edit operations ignores the influence of label set and the weight of edges. Therefore, we partly redefine these two edit operations as follows.

Vertex substitution Generally, the cost of a vertex substitution from v_1 to v_2 is equal to the string edit distance between the two vertices' labels. However, in TAGER, a vertex's label in a T-labeled graph may be consisted of several transitions' labels which is a label set. We take the label set into account when we compute the cost of vertex substitution.

$$\begin{aligned} cost(v_1, v_2)^{sub} = & \frac{k \cdot abs(|v_1.l| - |v_2.l|)}{|v_1.l| + |v_2.l|} \\ & + \frac{|v_1.l \cup v_2.l| - |v_1.l \cap v_2.l|}{|v_1.l \cup v_2.l|} \end{aligned} \tag{1}$$

$v_1.l$ and $v_2.l$ are the label sets of vertex v_1 and v_2 respectively, and $abs(x)$ represents the absolute value of x. k is a user-defined factor used to weight the effect by scale difference, as the experimental best value we set k to 2. The first part of Equation (1) represents the scale difference of two vertices. As conflict/parallel structure will bring labels of multiple transitions into one vertex, the label set of a vertex may contain more than one label. The second part is the pure substitution cost. This part of cost is influenced by two vertices' label sets, which is denoted as the difference between the two label sets. We exemplify the soundness of the substitution operation with two examples on Petri net for directly perceived feeling. Readers can compute the T-labeled graphs according to Definition 5 instead.

Fig. 4. Simple case of vertex substitution

Example 1. Figure 4 shows that if the scales of the vertices are equal, the upper bound of the substitution cost will be one. Converting S_1 to S_{1a} and S_2 to S_{2a} have the same cost.

Fig. 5. Complex case of vertex substitution

Example 2. In Figure 5, we present a complex situation when the scales of the vertices are different. Converting S_3 to S_{3a}, S_3 to S_{3b} and S_3 to S_{3c} requires a substitution, an insertion and a deletion of a vertex respectively in the original Petri net. However, only one vertex substitution occurs in the corresponding T-labeled graphs, from vertex label set $\{T_1, T_2, T_3\}$ to $\{T_1, T_4, T_3\}$, $\{T_1, T_2, T_3, T_4\}$ and $\{T_1, T_2\}$ respectively. The costs of the three substitutions will be $0+2/4+0 = 1/2$, $2/7 + 1/4 + 0 = 15/28$ and $2/5 + 1/3 + 0 = 11/15$ respectively.

Compared to Figure 4, we can see that a partial substitution costs less than a whole one. Besides, if the reference model is unchanged, a deletion affects their similarity bigger than an insertion. This shows that our vertex substitution can distinguish all general situations.

Edge insertion/deletion In most situations, the cost of edge insertion/deletion is assigned one. In our approach, we assign weight to an edge according to its type and structure in a T-labeled graph. Firstly, we determine if an edge e is a *loopedge* (edges related to the loop structure) or a *skipedge* (edges related to the conflict structure). If so, the weight of e is set as the max span of the sequential part from source place to sink place of the T-labeled graph where e is repeated or skipped over, denoted as $span_{loop}(e)$ or $span_{conflict}(e)$. Otherwise, it is set as one. Therefore, the cost of inserting or deleting an edge e is defined as:

$$cost(e)^{ins/del} = max(span_{loop}(e), span_{conflict}(e), 1) \tag{2}$$

For the given model in Figure 3, the third vertex pair $(\{T_1\}, \{T_1, T_3\})$ requires a substitution operation. The cost of scale difference part is $2 \times (2-1)/(2+1) = 0.667$, while the cost of node label set difference part is $(2-1)/2 = 0.5$. As we take into account the conflict span only when its scale is more than one or it is a general edge, the cost of conflict or loop part is 0. Therefore, their substitution cost is $0.667 + 0.5 = 1.167$.

3.4 The graph edit distance

One graph can be converted to another using the edit operations described above. If we assign appropriate weight to the cost of each operation, we can get the total cost of converting one graph to another, which is so-called graph edit distance.

Definition 6. ***(Graph edit distance)*** *Let $G_1 = (N_1, E_1, \lambda_1)$ and $G_2 = (N_2, E_2, \lambda_2)$ be two T-labeled graphs. Let $M : N_1 \nrightarrow N_2$[1] be a function that maps nodes in G_1 to nodes in G_2. Let $dom(M) = \{n_1 \in N_1 | (n_1, n_2) \in M\}$ be the domain of M and $cod(M) = \{n_2 \in N_2 | (n_1, n_2) \in M\}$ be the codomain of M. Given $n \in N_1 \cup N_2$, n is substituted iff $n \in dom(M)$ or $n \in cod(M)$. SubNodes is the set of all substituted nodes. A node $n_1 \in N_1$ that is deleted from G_1 or inserted in G_1 iff it is not substituted. SkipNodes is the set of all inserted and deleted nodes in G_1. SkipEdges is the set of all inserted and deleted edges in G_1. Let wsubn, wskipn and wskipe be the weights that we assign to substituted nodes, inserted or deleted nodes, inserted or deleted edges respectively.*

The graph edit distance induced by the mapping M is defined as:

$$\begin{aligned} editDistance(G_1, G_2) = & wskipn \cdot |SkipNodes| + wskipe \cdot |SkipEdges| \\ & + wsubn \cdot |SubNodes| \end{aligned}$$

$editDistance(G_1, G_2)$ represents the graph edit distance between two T-labeled graphs G_1 and G_2. As the experimental best performance, we take $k = 2.0$, $wskipn = 0.8$, $wsubn = 1.8$, $wskipe = 0.5$. And all these parameters can be defined by users. For example, we compute the graph edit distance of the two graphs in Figure 3. To convert T-labeled graph M_0 to M_4, we need to substitute $\{T_1\} \to \{T_1, T_3\}$. The minimal graph edit distance between M_0 and M_4 in Figure 3 is induced by the following mapping:

$$Mapping : \{(\{T_0\}, \{T_0\}), (\{T_1\}, \{T_1, T_3\}), (\{T_2\}, \{T_2\})\}$$

[1] We use $\nrightarrow$ to represent the partial injective mapping.

The graph edit distance between M_0 and M_4 is (2/3+1/2)*1.8 = 2.1. We use the Greedy algorithm [**?**] to guarantee that the graph edit distance between the two graphs is the minimal possible distance.

3.5 The similarity computation

Both graph edit distance and the scale of graphs should be considered when we compute the similarity. Thus, we define the graph similarity as follows.

Definition 7. ***(Graph similarity)*** *Let $G = (V, E, \lambda)$ represents the T-labeled graph we get from the previous step. $|G|$ is the scale of T-labeled graphs G, which is the total sum of $|V|$ and $|E|$. The similarity measure of two T-labeled graphs G_1 and G_2 is defined as follows:*

$$Similarity(G_1, G_2) = 1 - \frac{editDistance(G_1, G_2)}{|G_1| + |G_2|} \tag{3}$$

The similarity can be viewed as the complement of differentiation. We can compute the differentiation by the edit distance, so we just need to consider the influence by model scale or graph scale.

3.6 The algorithm analysis of TAGER

We provide an integrated algorithm (see Algorithm 1) in this part, which basically follows the steps as we mentioned in Section 3.1.

Algorithm 1 TAGER algorithm

```
Input Two labeled Petri nets $N_1 = (P_1, T_1, F_1, M_{01}, A, l_1)$, $N_2 = (P_2, T_2, F_2, M_{02}, A, l_2)$
Output The similarity measure of $N_1$ and $N_2$
  $CoverabilityG_1 \leftarrow CoverabilityGraphBuilder(N_1)$
  $CoverabilityG_2 \leftarrow CoverabilityGraphBuilder(N_2)$
  Use DFS to go through the whole graph to store the transition labels on all edges to the successive vertices.
  T-labeled$G_1 \leftarrow convert(CoverabilityG_1)$
  T-labeled$G_2 \leftarrow convert(CoverabilityG_2)$
  Use the Remco's greedy algorithm [?] to compute the edit distance between two graphs with the edit operations and costs redefined in this paper.
  $editdis \leftarrow editDistance($T-labeled$G_1$, T-labeled$G_2)$
  $similarity \leftarrow 1 - \frac{editdis}{|\text{T-labeled}G_1| + |\text{T-labeled}G_2|}$
  return $similarity$
```

The greedy algorithm part in TAGER, which can be used to compute the graph edit distance, is in $O(n^3)$ time complexity where n is the number of nodes of the largest graph.

As we consider all the possible factors that will influence the measure so that we can provide a precision similarity measure value. The time complexity of TAGER is $O(|V_1|^3 + |V_2|^3)$, $|V|$ is the number of vertices in a T-labeled graph.

The T-labeled graph will unfold the parallel structure. Although they will converge in the end, there is possibility of state explosion. For the same reason, our algorithm takes a longer time when the computed model contains a parallel structure with many branches.

4 Experimental evaluation of TAGER

We have implemented TAGER in BeehiveZ[2] which is a business process data management system developed by Tsinghua University. In this section, we present the experimental evaluation of TAGER in both performance and quality perspectives. We evaluate the performance by applying TAGER on real-life dataset and evaluate the quality by comparing TAGER with other algorithms.

4.1 Experimental setting

Our experimental datasets come from two parts, the artificial process models and the TC, DG and SAP reference models. These collections has 578 models (expressed in Petri nets) in total. The artificial models are mainly used to check whether TAGER can satisfy the properties that a good business process similarity measure should meet. The models coming from TC, DG and SAP can check whether TAGER meets the distance metric requirement.

The basic features of these models are summarized in Table 1. The following paragraphs briefly introduce the sources of the datasets.

- *The DG Dataset.* We have collected 94 real-life process models from Dongfang Boiler Group Co., Ltd. Dongfang Boiler is a major supplier of thermal power equipment, nuclear power equipment, power plant auxiliaries, chemical vessels, environmental protection equipment and coal gasification equipment in China (with more than 33% market share).
- *The TC Dataset.* We have collected 89 real-life process models from Tangshan Railway Vehicle Co., Ltd. (TRC). TRC is the oldest enterprise in Chinas railway transportation equipments manufacturing industry, with its predecessor constructing China's first railway line from Tangshan to Xugezhuang in 1881 during the westernization movement at the end of the Qing Dynasty. Today the company has built the train that has achieved the highest speed of Chinese railways - 394.3km/h.
- *The SAP Dataset.* SAP dataset is consist of SAP reference models. We delete such models that cannot satisfy the soundness property of WF-net. Then the dataset scale is decreased from 604 to 389.

[2] https://code.google.com/p/beehivez/

Table 1. The features of three real-life reference model sets

Dataset	Size	Average			Minimum			Maximum		
		#transitions	#places	#arcs	#transitions	#places	#arcs	#transitions	#places	#arcs
SAP	389	4.465	7.504	11.514	1	2	2	21	31	56
DG	94	8.553	8.883	17.777	1	3	3	34	33	70
TC	89	11.472	10.281	22.933	6	5	11	28	29	58

We repeatedly extract every three models from the dataset, calculate the similarities between each pair of them and test whether they meet the triangle inequality property. The following two equations [**?**] are used to measure the triangle inequality property.

Definition 8. ***(Triangle Inequality Property)*** *For any three models in one dataset, e.g.* N_0, N_1, N_2*, the distances between any two of them are* $dis(N_0, N_1)$, $dis(N_0, N_2)$, $dis(N_1, N_2)$ *respectively, the triangle inequality property requires these distances to satisfy that the sum of any two of them should be bigger than the third one.*

$$dis(N_0, N_1) = 1 - Similarity(N_0, N_1) \tag{4}$$

$$dis(N_0, N_1) = \begin{cases} \infty & Similarity(N_0, N_1) = 0 \\ 1/Similarity(N_0, N_1) - 1 & \text{otherwise} \end{cases} \tag{5}$$

The range of TAGER's similarity value is $[0, 1]$, and the distances of those two equations both decrease with the increasing similarity value. The range of Equation (4) is $[0, 1]$, and the range of Equation (5) is $[0, \infty)$. As if there is more than one ∞ among the three distances, the triangle property should be satisfied.

4.2 Performance evaluation on real-life models

Table 2. The performance of TAGER

-	average time1 (ms)[1]	average time2 (ms)[2]	satisfaction Eq.(4)[3]	satisfaction Eq.(5)[4]
SAP	4	3	1.0	0.95
DG	8	21	1.0	0.86
TC	11	33	1.0	0.96

[1] the average time spent on building the T-labeled graph for each process model in dataset.

[2] the average time spent on computing the similarity between two T-labeled graphs.

[3] the satisfaction rate of triangle inequality property using Equation (4).

[4] the satisfaction rate of triangle inequality property using Equation (5).

The performance of TAGER on real-life datasets are shown in Table 2. As transitions, places and arcs of models in the TC dataset is far more than those models in the other two datasets, the average time spent on building coverability graph and computing similarity value on the TC dataset are the highest of all. We can conclude that time consuming on a dataset is strongly related to the complexity of models in the dataset.

Moreover, TAGER on the three datasets has a 100% satisfaction on the triangle inequality property using Equation (4) while it performs well enough on the satisfaction using Equation (5).

Besides, TAGER has the 0.115ms average calculating time on 10,000 artificial models which are generated by BeehiveZ.

4.3 Quality evaluation on artificial models

Fig. 6. Artificial models from M_0 to M_8

We evaluate TAGER by satisfying the properties that a good similarity algorithm should meet. The former six properties have been proposed in [**?**], we

Fig. 7. Artificial models from M_9 to M_{16}

just use them directly. Besides, we propose a new property named parallel structure drift invariance. We provide the following models as shown in Figure 6 and Figure 7 to verify these properties. We compare TAGER with other algorithms such as TAR [?], PTS [?], SSDT [?], BP [?] and CF [?] which have a good performance on these properties, too. The artificial dataset (total 17 models) are collected from [?,?] and we have manually modified them slightly.

- **TAR**: Zha et al. [?] discuss a naive similarity measure, also called reference similarity, based on the set of transition adjacency relations.
- **PTS**: Wang et al. [?] propose a similarity measure based on principal transition sequences.
- **SSDT**: Wang et al. [?] define a shortest succession distance between tasks to compute the similarity value of two Petri nets.
- **BP**: Weidlich et al. [?] capture the behavior of a process model by examining dependencies between the execution of transitions.
- **CF**: Dijkman et al. [?] use causal footprints for capturing the precedence relations between transitions in a process model.

Property 1. **Sequential structure drift invariance**
If a Workflow net is sound, no matter where we add a sequential structure to the original sequential structure to derive new process models, they have the same similarity with the original one. The similarity should keep unchanged regardless the position of the added sequential structure.

For example, M_1, M_2, M_3 are derived from M_0 by adding transition T_3. When these three models are compared with M_0, they should has the same similarity value. The evaluation results on property 1 are shown in Table 3. We can see that except TAR, the other algorithms can satisfy this property.

Table 3.
The evaluation results on property 1

sim	(M_0, M_1)	(M_0, M_2)	(M_0, M_3)
TAR	0.250	0.250	0.667
PTS	0.750	0.750	0.750
SSDT	0.813	0.813	0.813
BP	0.450	0.450	0.450
CF	0.286	0.286	0.286
TAGER	0.816	0.816	0.816

Table 4.
The evaluation results on property 2

sim	(M_0, M_4)	(M_0, M_5)
TAR	0.500	0.667
PTS	0.889	0.778
SSDT	0.750	0.688
BP	0.460	0.525
CF	0.334	0.353
TAGER	0.873	0.800

Property 2. **Negative correlation by conflict span**
This property means that when modifying a model by adding a conflict structure on the sequential part, the longer the conflict span is, the greater the influence it makes on the similarity between the new model and the original one. The negative correlation reflects the special structure affecting the similarity.

For example, M_4 is more similar to M_0 than M_5. The evaluation results on property 2 are shown in Table 4. We can see that except TAR, BP and CF, all others algorithms satisfy this property.

Property 3. **Unrelated task regression**
This property means the similarity between two models will decrease by adding more unrelated tasks.

For example, compare the similarity of M_0 and M_6, M_7, M_8 separately. M_6 replaces transition T_2 by T_3, M_7 adds T_4 as a conflict structure at the same position and M_8 adds one more T_5. The similarity between M_0 and M_6, M_7, M_8 should decrease. The specific result in Table 5 shows that except PTS, all other algorithms satisfy property 3. However, the results of TAR and BP are too low, the values of TAGER are close to the desirable ones. Besides, we prove that adding unlimited conflict structures will bring a limited influence on the similarity, and the lower bound is 0.672. It is a reasonable value for this situation.

Table 5.
The evaluation results on property 3

sim	(M_0, M_6)	(M_0, M_7)	(M_0, M_8)
TAR	0.250	0.167	0.125
PTS	0.667	0.667	0.667
SSDT	0.750	0.680	0.611
BP	0.220	0.143	0.100
CF	0.392	0.340	0.285
TAGER	0.891	0.818	0.782

Table 6.
The evaluation results on property 4

sim	(M_3, M_9)	(M_3, M_{10})
TAR	0.600	0.600
PTS	0.563	0.563
SSDT	0.760	0.720
BP	0.526	0.430
CF	0.259	0.242
TAGER	0.869	0.850

Property 4. **Negative correlation by loop length**
This property means that adding a loop structure in the sequential structure, the similarity will decrease with the increasing scale of the loop structure.

For the models shown in Figure 6 and 7, the similarity between M_3 and M_9 is bigger than that between M_3 and M_{10}. The specific result in Table 6 shows that except TAR and PTS, all others algorithms satisfy this property.

Property 5. **Conflict structure drift invariance**
This property means if adding a conflict structure in the sequential part, no matter where the conflict structure is added in the sequential structure, its influence to the similarity between the new model and the original model should keep unchanged.

For the models shown in Figure 6 and 7, the similarity between M_3 and M_{11}, M_3 and M_{12}, M_3 and M_{13} should be the same. The specific result in Table 7 shows that except TAR and CF, all others algorithms satisfy this property

Table 7. The evaluation results on property 5

sim	(M_3, M_{11})	(M_3, M_{12})	(M_3, M_{13})
TAR	0.600	0.600	0.750
PTS	0.917	0.917	0.917
SSDT	0.800	0.800	0.800
BP	0.514	0.514	0.514
CF	0.247	0.247	0.252
TAGER	0.900	0.900	0.900

Property 6. **Loop structure drift invariance**
This property means if adding a loop structure in the sequential part, no matter where the loop structure is added in the sequential structure, its influence to the similarity between the new model and the original model should keep unchanged.

For the models shown in Figure 6 and 7, the similarity between M_3 and M_9, M_3 and M_{14} should be the same. The specific result in Table 8 shows that except PTS, all other algorithms satisfy this property.

Table 8.
The evaluation results on property 6

sim	(M_3, M_9)	(M_3, M_{14})
TAR	0.600	0.600
PTS	0.563	0.625
SSDT	0.760	0.760
BP	0.526	0.526
CF	0.259	0.259
TAGER	0.869	0.869

Table 9.
The evaluation results on property 7

sim	(M_3, M_{15})	(M_3, M_{16})
TAR	0.429	0.429
PTS	0.800	0.800
SSDT	0.760	0.760
BP	0.547	0.547
CF	0.226	0.226
TAGER	0.766	0.766

Property 7. **Parallel structure drift invariance**
This is the new property we proposed in this paper. It means if adding a parallel structure in the sequential part, no matter where the parallel structure is added in the sequential structure, its influence to the similarity between the new model and the original model should keep unchanged.

For the models shown in Figure 6 and 7, the similarity between M_3 and M_{15}, M_3 and M_{16} should be equal. The specific result in Table 9 shows that all algorithms satisfy this property.

Table 10 gives an overview of the quality evaluation on existing algorithms. It is obvious that TAGER and SSDT are the best ones. However, SSDT depends on another process model to compute the SSDT matrix, so it cannot be used as a kind of process model index.

Table 10. Summary of the quality evaluation

-	Property1	Property2	Property3	Property4	Property5	Property6	Property7
TAR	×	×	✓	×	×	✓	✓
PTS	✓	✓	×	×	✓	×	✓
SSDT	✓	✓	✓	✓	✓	✓	✓
BP	✓	×	✓	✓	✓	✓	✓
CF	✓	×	✓	✓	×	✓	✓
TAGER	✓	✓	✓	✓	✓	✓	✓

5 Conclusion and Outlook

We provide a simple algorithm to calculate similarity between process models, which is based on the T-labeled graphs of Petri nets. As the graph edit distance is a classical approach to measure the difference between two graphs, we choose the edit distance as the crucial factor. The traditional operations work effectively in the situation containing no loop/conflict structures. We extend the substitution operation by considering the label sets of vertices and the cost of the loop/conflict structure. Experimental results show that TAGER has a good performance and meets all the properties while most other main-stream algorithms cannot. As parameters play an important role in TAGER and they are required to change frequently when datasets vary, we will focus on how to make the adjustment of parameters more automatically. Moreover, designing a new process model index based on TAGER is also part of our future work.

Acknowledgement

The research is supported by the MOE-CMCC research foundation project No. MCM20123011 and the special fund for innovation of Shandong,China No. 2013CXC30001.

References

1. Tadao Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.