Fast Business Process Similarity Search with Feature-Based Similarity Estimation

Zhiqiang Yan, Remco Dijkman, and Paul Grefen

Eindhoven University of Technology
P.O. Box 513, NL-5600 MB Eindhoven, The Netherlands
{z.yan,r.m.dijkman,p.w.p.j.grefen}@tue.nl

Abstract. Nowadays, business process management plays an important role in the management of organizations. More and more organizations describe their operations as business processes, and the intra- and inter-organizational interactions between operations as services. It is common for organizations to have collections of hundreds or even thousands of business processes. Consequently, techniques are required to quickly find relevant business process models in such a collection. Currently, techniques exist that can rank all business process models in a collection based on their similarity to a query business process model. However, those techniques compare the query model with each model in the collection in terms of graph structure, which is inefficient and computationally complex. Therefore, this paper presents a technique to make this more efficient. The technique selects small characteristic model fragments, called features, which are used to efficiently estimate model similarities and classify them as *relevant, irrelevant or potentially relevant* to a query model. Only *potentially relevant* models must be compared using the existing techniques. Experiments show that this helps to retrieve similar models at least 3.5 times faster without impacting the quality of the results; and 5.5 times faster if a quality reduction of 1% is acceptable.

1 Introduction

Nowadays, service and business process management technologies develop quickly in both academic and industrial fields. To increase the flexibility and controllability of the management of organizations, business processes are used to describe functions organization provides and services are used to describe the both intra- and inter-organizational cooperations. As a result, it is common to see collections of hundreds or even thousands of business process models. For example, the SAP reference model consists of more than 600 business process models [16], and the reference model for Dutch Local Governments contains a similar number of models [9]. As business process model collections increase in size, tools and techniques are required to manage them. This includes tools and techniques for quickly searching a collection for business process models that meet certain criteria. These criteria can be specified by means of a search query [2,5], but also by means of (a part of) a business process model for which similar models must be retrieved [6,7].

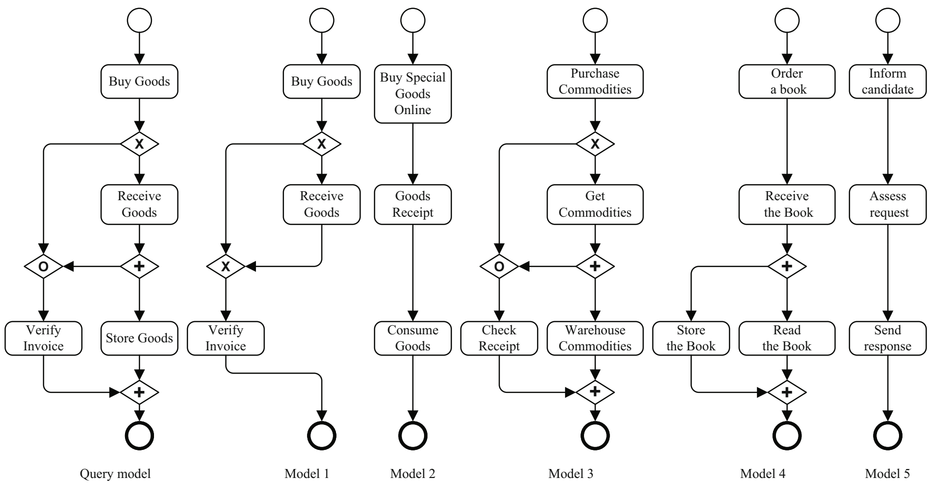

Fig. 1. Searching a collection of business process models

This paper focuses on the second class of search techniques, which are also called *similarity search* techniques. Similarity search can, for example, be applied to search a collection of reference process models for the model that best matches a process model from a specific organization; or in case of merger between two organizations to search which business process model from one organization matches which business process model of the other. Figure 1 shows an example of a business process similarity search. It shows one *query model* and five *process models* in the BPMN notation. Given a search query model, a similarity search technique should only return those process models that are similar to the search query model and it should return those similar process models in order of their similarity to the search query model. In the example, the technique could return models 1, 2 and 3.

There currently exist similarity search techniques [6,7]. However, these techniques focus on defining a metric to compute the similarity between two process models. To rank the business process models in a collection, the similarity of each of the process models to the query model must be computed. Subsequently, the process models are ordered according to their similarity. This is time consuming and can cause a similarity search operation to take multiple seconds or even minutes, depending on the metric and algorithm that is used, while a search query should be performed within milliseconds by a search engine, e.g., Google.

Therefore, the goal of this paper is to develop a similarity search technique that is both accurate *and* fast. The technique works by quickly classifying process models as 'relevant', 'irrelevant' or 'potentially relevant' to a search query model, based on an estimation of their similarity to the search model. Existing similarity search techniques then only have to be used to rank the process models in the 'potentially relevant' category, which typically contains much fewer models than the collection as a whole (in our evaluation set only 10% of the number of models in the collection), therewith significantly reducing the search time.

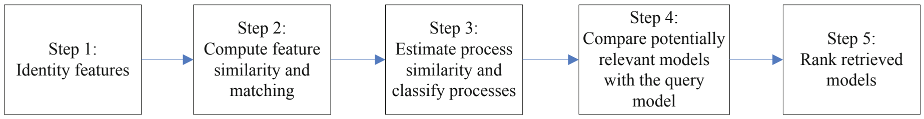
Step 1:
Identity features
Step 2:
Compute feature similarity and matching
Step 3:
Estimate process similarity and classify processes
Step 4:
Compare potentially relevant models with the query model
Step 5:
Rank retrieved models

Fig. 2. Steps of the technique in this paper

The technique classifies process models based on the number of *features* that they have that match with the search model. The technique consists of five steps, as shown in Figure 2. First, features are identified in the process models that have to be searched. Features are simple but representative abstractions of a business process model, e.g., tasks and task succession. Second, the search model and the process models are compared based by looking at the features that they have in common, i.e.: that are similar enough such that we say that they are 'matched'. For example, suppose that we use tasks and task succession as features in figure 1. We can observe that model 1 has six matching features with the search model: the task features 'Buy Goods', 'Receive Goods' and 'Verify Invoice'; and the succession features ('Buy Goods','Receive Goods'), ('Buy Goods','Verify Invoice') and ('Receive Goods','Verify Invoice'). Models 2 and 3 have fewer matches or have weaker similarity with respect to their matches (e.g.: 'Buy Goods', 'Buy Special Goods Online' and 'Purchase Commodities' are similar but not identical labels). Models 4 and 5 have an even weaker match or no match at all with respect to the features that they have in common with the search model. Third, an estimation of the similarity of process models to the search model is made, based on the ratio of matching features, and models are classified based on their estimated similarity. For example, depending on the exact metrics that are used, model 1 could be considered as relevant based on matching features, models 2 and 3 considered as potentially relevant, and models 4 and 5 as irrelevant. Fourth, using existing technologies [6], process model similarities are computed for potentially relevant models, e.g., models 2 and 3. Fifth, retrieved models are ranked. Relevant models, which are ranked by their *estimated* similarity, (e.g., model 1), are followed by potentially relevant models, which are ranked by the similarities computed in step 4 (e.g., model 2 and 3). This finally leads to a ranked list of search results.

Experiments show that the technology helps to retrieve similar models at least 3.5 times faster without impacting the quality of the results; and 5.5 times faster if a quality reduction of 1% as a tradeoff is acceptable.

The rest of the paper is organized as follows. Section 2 defines the concept of feature and presents features that can be used for business process similarity estimation. Section 3 defines metrics for measuring the similarity of features and checking whether features match. Section 4 presents metrics to determine whether a model is relevant, irrelevant or potentially relevant to a search query, based on the features that match with features from the query model. Section 5 presents the experiments to determine the search time and quality of the similarity search technique. Section 6 presents related work and section 7 concludes.

2 Business Process Model Features

In this paper features are defined as simple but representative abstractions of business process models. Their simplicity allows similarity computation based on them to be fast and their representativeness ensures that their similarity is strongly related to similarity of the business process models themselves. This makes features very suitable as means to quickly estimate the similarity of business process models.

Provided that we choose business process model features carefully, we can further speed up similarity search by building an index of business process models based on those features. In this section, we present the business process model features that we explore in this paper.

In previous work [6,7] we have shown that the most representative features of business process models are their task labels and their structure. This means that if two business processes have similar labels for their activities, it is also likely that the processes themselves are similar. Similarly, if two business processes have a similar structure, it is also likely that the processes themselves are similar.

Labels can be conveniently used as features, because they are simple strings and therefore qualify as simple abstractions. In addition, indexing mechanisms for strings are well-known, which enables indexing of label features. However, it is harder to use the structure of a business process model as a feature. In fact, considering the structure of a graph when computing the similarity between business process models in our previous work is what makes the problem computationally hard. Therefore, we consider the structure of a business process model in terms the more simple structural features: start, stop, sequence, split and join. We define these features on the abstraction of a business process graph.

Definition 1 (Business Process Graph, Pre-set, Post-set). *Let $\mathcal{L}$ be a set of labels. A business process graph is a tuple (N, E, λ), in which:*

- *N is the set of nodes;*
- *$E \subseteq N \times N$ is the set of edges; and*
- *$\lambda : N \to \mathcal{L}$ is a function that maps nodes to labels.*

Let $G = (N, E, \lambda)$ be a business process graph and $n \in N$ 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.

A business process graph is a graph representation of a business process model. As such, it is an abstraction of a business process model that focuses purely on the structure of that model, while abstracting from other aspects, e.g., different types of process modeling notations (BPMN, EPC, Petri net, etc.). However, because our measure of similarity is defined on the structure of a business process model, abstracting from these aspects is acceptable. Optionally, certain types of nodes can be disregarded in a business process graph. For example, figure 3 shows the business process graphs for the models from figure 1. Only tasks are considered in these graphs. Events and gateways are disregarded. We define our

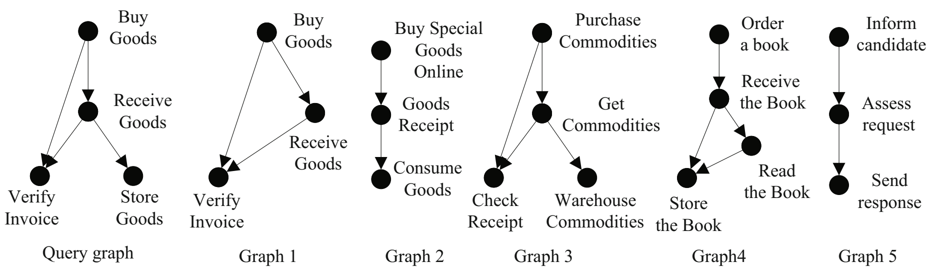

Fig. 3. Business process graphs

similarity search techniques on business process graphs to be independent of a specific notation.

Based on this the structural features are defined in Definition 2.

Definition 2 (Structural Business Process Model Features). *Let $G = (N, E, \lambda)$ be a business process graph.*

- *A start feature is a node $n \in N$ that has an empty pre-set;*
- *A stop feature is a node $n \in N$ that has an empty post-set;*
- *A sequence feature of size s is a list of nodes $[n_1, n_2, n_3, \ldots, n_s] \subseteq N$, such that $(n_1, n_2) \in E, (n_2, n_3) \in E, \ldots, (n_{s-1}, n_s) \in E$, for $s \geq 2$;*
- *A split feature of size s is a split node n and a set of nodes $\{n_1, n_2, \ldots, n_{s-1}\} \subseteq N$, such that $(n, n_1) \in E, (n, n_2) \in E, \ldots, (n, n_{s-1}) \in E$, for $s \geq 3$;*
- *A join feature of size s is a join node n and a set of nodes $\{n_1, n_2, \ldots, n_{s-1}\} \subseteq N$, such that $(n_1, n) \in E, (n_2, n) \in E, \ldots, (n_{s-1}, n) \in E$, for $s \geq 3$;*

For example, for graph 1 in figure 3, the label feature set is {Buy Goods, Receive Goods, Verify Invoice}, the start feature set is {Buy Goods} (using node labels to identify nodes), the stop feature set is {Verify Invoice}, the sequence feature set is {(Buy Goods, Receive Goods), (Buy Goods, Verify Invoice), (Receive Goods, Verify Invoice)}, the split feature set is {(Buy Goods,{Receive Goods, Verify Invoice})}, and the join feature set is {({Buy Goods, Receive Goods},Verify Invoice)}.

Many more possible features can be considered in business process models, depending on the business process model aspects that are taken into account (e.g. the organizational aspect or the data aspect), the desired performance of the algorithm (adding more features decreases performance) and the desired quality of the results (adding more features is expected to increase the quality of the search results). In this paper we focus on the most basic process model features. Extensions are possible and are a topic for future work.

Structural features can be considered in two fundamentally different ways:

1. by focusing on a single node and the interconnections that it has; or
2. by focusing on a small number of connected nodes.

More explanation will be given in Section 3.

3 Feature Similarity, Matching and Indexing

It is possible to use the similarity of the features of two business process models as an estimator of the similarity of the business process models themselves. To this end, metrics must be defined that quantify the similarity of the business process model features. We say that two features that are sufficiently similar are *matching features* and we show how we can determine feature matching based on their similarity. The ratio of matching features will be used in the next section as an estimator of the similarity of business process models. To be able to quickly identify matching features, and therewith similar business process models, feature indices must be defined.

This section first presents metrics to quantify feature similarity. Second, it explains how a feature match can be determined based on feature similarity and, third, it presents feature-based indices.

3.1 Feature Similarity

Label feature similarity can be measured in a number of different ways [7]. For illustration purposes we will use a syntactic similarity metric, which is based on string edit-distance, in this paper. However, in realistic cases more advanced metrics should be used that take synonyms and stemming [7] and, if possible, domain ontologies into account [10]. The label feature similarity is defined in the former work [6].

Definition 3 (Label Feature Similarity). *Let $G = (N, E, \lambda)$ be a business process graph and $n, m \in N$ be two nodes and let $|x|$ represent the number of characters in a label x. The string edit distance of the labels $\lambda(n)$ and $\lambda(m)$ of the nodes, denoted* $\mathrm{ed}(\lambda(n), \lambda(m))$ *is the minimal number of atomic string operations needed to transform $\lambda(n)$ into $\lambda(m)$ or vice versa. The atomic string operations are: inserting a character, deleting a character or substituting a character for another. The label feature similarity of $\lambda(n)$ and $\lambda(m)$, denoted* $\mathrm{lsim}(n, m)$ *is:*

$$\mathrm{lsim}(n, m) = 1.0 - \frac{\mathrm{ed}(\lambda(n), \lambda(m))}{\max(|\lambda(n)|, |\lambda(m)|)}$$

For example, the string edit distance between ‘Transportation planning and processing’ and ‘Transporting’ is 26: delete ‘ion planning and process’. Consequently, the label feature similarity is $1.0 - \frac{26}{38} \approx 0.32$. Optional pre-processing steps, such as lower-casing and removing special characters, can improve the results of feature similarity measurements.

The drawback of measuring feature similarities only by labels is that related tasks can be labeled differently. For example, in figure 3, ‘Buy Special Goods Online’ of Graph 2 and ‘Purchase Commodities’ of Graph 3 are related to ‘Buy Goods’ of Query graph. However, compared with ‘Buy Goods’, the former one is more verbose and the later one uses synonyms. They may not match based on the label similarity. To deal with this situation, we use structural information together with the labels.

We can measure the structural similarity of two nodes, by determining the similarity of the (structural) roles that they have in their business process graphs. We distinguish five different roles that nodes can have: start, stop, sequence, split or join. We do not distinguish the type of splits or joins (e.g.: XOR or AND), because we established in previous work [6] that the similarity of the types of two splits or two joins is a bad indication for whether they are similar. Although we ackowledge that the disctinction between different types of splits or joins is of utmost importance when determining behavioral equivalence, we point out that measuring similarity is different from determining equivalence.

Definition 4 (Role Feature). *Let* $n \in N$ *be a node and* $\mathcal{R} = \{\text{start}, \text{stop}, \text{split}, \text{join}, \text{regular}\}$ *be a set of roles that a node can have. The roles of* n *are determined by the function* $\text{roles} : N \to I\!P(\mathcal{R})$*, such that*

$$\begin{array}{ll}
\text{start} \in \text{roles}(n) & \Leftrightarrow |\bullet n| = 0 \\
\text{stop} \in \text{roles}(n) & \Leftrightarrow |n \bullet| = 0 \\
\text{split} \in \text{roles}(n) & \Leftrightarrow |n \bullet| \geq 2 \\
\text{join} \in \text{roles}(n) & \Leftrightarrow |\bullet n| \geq 2 \\
\text{regular} \in \text{roles}(n) & \Leftrightarrow |\bullet n| = 1 \wedge |n \bullet| = 1
\end{array}$$

Roles of nodes are considered to be similar or not with respect to the input and output paths of the nodes. The definition of role feature similarity is inspired by string edit-distance, i.e., mainly considering the differences between numbers of input (output) paths of two nodes. Formally, role feature similarity is defined as follows:

Definition 5 (Role Feature Similarity). *Let* $n, m \in N$ *be two nodes. The role feature similarity of these two nodes, denoted* $\text{rsim}(n,m)$*, is defined as:*[1]

$$\text{rsim}(n,m) = \begin{cases}
1 & \textit{if } \text{start} \in \text{croles} \wedge \text{stop} \in \text{croles} \\
\text{avg}(1 - \frac{\text{abs}(|n\bullet| - |m\bullet|)}{|n\bullet| + |m\bullet|}, 1) & \textit{if } \text{start} \in \text{croles} \wedge \text{stop} \notin \text{croles} \\
\text{avg}(1, 1 - \frac{\text{abs}(|\bullet n| - |\bullet m|)}{|\bullet n| + |\bullet m|}) & \textit{if } \text{start} \notin \text{croles} \wedge \text{stop} \in \text{croles} \\
\text{avg}(1 - \frac{\text{abs}(|n\bullet| - |m\bullet|)}{|n\bullet| + |m\bullet|}, & \\
\quad 1 - \frac{\text{abs}(|\bullet n| - |\bullet m|)}{|\bullet n| + |\bullet m|}) & \textit{otherwise}
\end{cases}$$

Where $\text{croles} = \text{roles}(n) \cap \text{roles}(m)$.

This formula covers all possible combinations of roles that nodes can have. For example, the situation in which both nodes are split nodes as well as join nodes is covered by the case 'otherwise' ($\text{start} \notin \text{croles} \wedge \text{stop} \notin \text{croles}$). The situation in which both nodes are sequence nodes is covered by the same case and leads to a role feature similarity score of 1.

The drawback of measuring role similarity in this way is that it does not discount for the fact that there is a large difference between the frequency of the occurrence of the different role features. Therefore, using the role similarity

[1] *avg* returns the average value; *abs* returns the absolute value.

metric in this way is ineffective. Since, if we give a bonus for matching role features, most nodes would receive that bonus. Therewith, the effect of the bonus would be minimal.

For that reason we refine the role similarity metric to take this effect into account. We do that by not considering features that appear too frequently in the dataset; we say that those features lack 'discriminative power'.

Definition 6 (Discriminative Role Features). *Let $\mathcal{N}$ be the set of nodes of all business process models in a collection. We say that a role feature $r \in \mathcal{R}$ is discriminative, denoted* discriminative(r) *if and only if the fraction of the nodes that have the feature is sufficiently small:*

$$\frac{|\{n|n \in \mathcal{N}, \mathrm{r} \in roles(n)\}|}{|\mathcal{N}|} \leq \mathrm{dcutoff}$$

Where dcutoff *is a cutoff value that determines when the fraction of nodes that have the feature is sufficiently small. This cutoff value is a parameter that can be set as desired, to produce the best results.*

In general a good setting for dcutoff is easy to determine, because there is a large difference between the frequency of features with a low frequency of occurrence and features with a high frequency of occurrence. For example, in the set of business process models that we use for evaluation in this paper, there are 374 nodes in total and 178 the 'stop' role, 153 have the 'start' role, 58 the 'seq' role, 52 the 'split' role, 36 the 'join' role. Here, we have far more nodes with the 'start' and 'stop' roles than other nodes. Hence, if we set the dcutoff anywhere between 0.16 and 0.40, 'start' and 'stop' role features are not considered discriminative, while other role features *are* considered discriminative. We incorporate the discriminative power of role features into their similarity using the following formula.

Definition 7 (Role Feature Similarity with Discriminative Power). *Let $n, m \in N$ be two nodes. Their role feature similarity with discriminative power, denoted* rdsim(n, m)*, is defined as:*

$$\mathrm{rdsim}(n,m) = \begin{cases} \mathrm{rsim}(n,m) & \textit{if}\ \forall \mathrm{r} \in \mathrm{roles}(n) \cap \mathrm{roles}(m) : \mathrm{discriminative(r)} \\ 0 & \textit{otherwise} \end{cases}$$

3.2 Feature Matching

We say that two features are matched if they are sufficiently similar. What is considered to be sufficient is determined by cutoff parameters that can be set accordingly. If two business process models have sufficiently many matching features, we consider them similar. This is explained in the next section.

We consider two node features to be matched, if their component features (label features and role features) are matched. Strong label feature similarity is a strong indication that two nodes are matched, while a combination of role feature similarity and (less strong) label feature similarity is also an indication that two nodes are matched. We distinguish between these two cases when determining a node feature match, such that we can set different thresholds for label similarity in case there is also role similarity and in case there is no role similarity.

Definition 8 (Node Feature Match). *Let $n, m \in N$ be two node features with their respective label features and role features. The node features are matched if they satisfy one of the following two rules:*

- *their label features are similar to a high degree, i.e.,* $\mathrm{lsim}(n, m) \geq \mathrm{lcutoff}_{\mathrm{high}}$*;*
- *their role features are similar, and their label features are similar to a medium degree, i.e.,* $\mathrm{rdsim}(n, m) \geq \mathrm{rcutoff}$ *and* $\mathrm{lsim}(n, m) \geq \mathrm{lcutoff}_{\mathrm{med}}$.

Where $\mathrm{lcutoff}_{\mathrm{high}}$, $\mathrm{rcutoff}$ *and* $\mathrm{lcutoff}_{\mathrm{med}}$ *are parameters that determine what is considered to be a similar to what degree. The parameters can be set as desired, to produce the best results.*

We consider two structural features to be matched, if their component features (node features) are matched.

Definition 9 (Structural Feature Match). *Two start features with nodes n and m are matched if and only if their node features are matched. A stop feature match is defined similarly.*

Two sequence features of size s with lists of nodes $Ln = [n_1, n_2, n_3, \ldots, n_s]$ and $Lm = [m_1, m_2, m_3, \ldots, m_s]$ are matched if and only if for each $1 \geq i \geq s$: the node features of n_i and m_i are matched.

Two split features of size s with split nodes n and m and sets of nodes $Sn = \{n_1,\ n_2,\ \ldots, n_{s-1}\}$ and $Sm = \{m_1, m_2, \ldots, m_{s-1}\}$ are matched if and only if the node features of nodes n and m are matched and there exists a mapping $\mathrm{Map} : Sn \to Sm$ *holds that for each* $(\mathrm{sn}, \mathrm{sm}) \in \mathrm{Map}$*: the node features of sn and sm are matched. A join feature match is defined similarly.*

Features of different types or sizes are never matched with each other. We can use these two definition to define general feature matching.

Definition 10 (Feature Match). *Let f_1 and f_2 be two features. f_1 and f_2 are matched, denoted* $\mathrm{match}(f_1, f_2)$*, if and only if they are of the same type and they are matched according to definition 8 in case they are node features or definition 9 in case they are structural features.*

3.3 Index Construction

Node feature matching is mainly based on label similarity and structural feature matching is as well, because it is based on node feature matching. Therefore, if we can speed up finding similar labels, we can speed up feature matching. We use two index techniques to speed up finding similar labels.

First, we use an M-Tree index [3] on node labels. An M-Tree index is specifically meant for quickly finding items that are similar to a given item to a given degree. In our case, we can use it to quickly find node labels that have a label feature similarity (definition 3) with a given node label that is higher than a specified cutoff (lcutoff$_{\text{high}}$ or lcutoff$_{\text{med}}$ in definition 8).

Second we use an inverted index [14] that maps node labels to nodes, to obtain the set of similar nodes from the set of similar labels. We use the inverted index, because multiple nodes with the same label may exist in a collection of business process models. For example, in the set of business process models that we use for evaluation in this paper, there are 374 labels, but only 190 distinct ones. The inverted index can prevent comparing identical labels repeatedly.

For other features, we also build inverted indices to prevent comparing identical ones repeatedly. Furthermore, we can build another index, if we exploit the fact that, to match a sequence of two nodes, we always need to match a node feature and, to match a sequence of three nodes, we always need to match a sequence of two nodes. In other words to match a 'larger' feature, we always need to match a 'smaller' feature. Using this observation we can build a parent/child index. This index functions like a cache that stores the similarity between smaller 'parent features' as well as the relation between larger 'child features' and their smaller 'parent features'. Using this cache, to match two 'child features' we can look up the corresponding 'parent features' and their similarity.

Definition 11 (Parent Feature, Child Feature). *If feature A can generate feature B by adding some node(s), feature A is a parent feature of feature B, and feature B is a child feature of feature A. If feature A can generate feature B by adding only one node, feature A is a direct parent feature of feature B, and feature B is a direct child feature of feature A.*

For example, node feature 'Buy Goods' is a direct parent feature of sequence feature ('Buy Goods', 'Receive Goods'), and sequence feature ('Buy Goods', 'Receive Goods') is a direct child feature of node feature 'Buy Goods'. An example of the parent/child feature index is shown in figure 4, which is an example of for graph 2 in figure 3. The index has different feature size levels, and the features of the same size are at the same level. Between levels features connect to their direct parent and child features. By doing this, we can start comparing features from node feature level and only need to compare larger features whose parent features are matched.

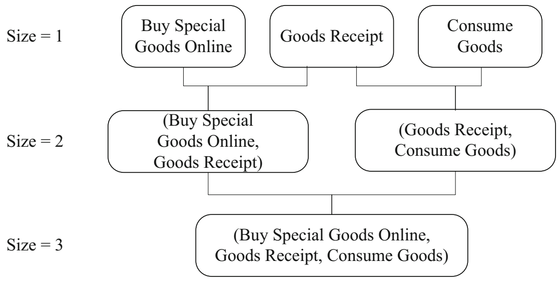
Size = 1
Buy Special Goods Online
Goods Receipt
Consume Goods
Size = 2
(Buy Special Goods Online, Goods Receipt)
(Goods Receipt, Consume Goods)
Size = 3
(Buy Special Goods Online, Goods Receipt, Consume Goods)

Fig. 4. Feature Index

4 Feature-Based Similarity Estimation

We use the fraction of features that match between two business process models to estimate their similarity, as shown in Definition 12.

Definition 12 (Estimated Business Process Model Similarity). *Given a query process graph G_q and another process graph G, with feature sets F_q and F derived automatically from G_q and G. The estimated business process similarity, denoted* $\mathrm{GSim}(G_q, G)$ *is the number of features in G_q or G that are matched by a feature in the other process graph, divided by the number of all features in G_q and G:*

$$\frac{|\{f_q \in F_q | \exists f \in F : \mathrm{match}(f_q, f)\}| + |\{f \in F | \exists f_q \in F_q : \mathrm{match}(f_q, f)\}|}{|F_q| + |F|}$$

Note that we count the number of features in G_q that match a feature in G separately from the number of features in G that match a feature in G_q, because the match is not necessarily one-to-one. For example, a label feature ‘Fill-out Request Forms’ can match with label features ‘Fill-out Requester’s Detail’ and ‘Fill-out Request Details’ in the other process graph.

Based on the estimated graph similarity, we can classify graphs as relevant, irrelevant or potentially relevant to a search graph. We do that by defining the minimal estimated similarity that a graph must have to the search query graph to be considered relevant and the minimal estimated similarity that a graph must have to be considered potentially relevant. We retrieve relevant one directly, check the potentially relevant ones with expensive similarity search algorithms [6,7], and discard irrelevant ones.

Definition 13 (Graph Relevance Classification). *Given a query process graph G_q and another process graph G, we classify G as:*

- *relevant to G_q if and only if* $\mathrm{GSim}(G_q, G) \geq \mathrm{ratio_r}$
- *potentially relevant to G_q if and only if* $\mathrm{ratio_r} > \mathrm{GSim}(G_q, G) > \mathrm{ratio_p}$
- *irrelevant to G_q if and only if* $\mathrm{ratio_p} \geq \mathrm{GSim}(G_q, G)$

Where $\mathrm{ratio_r}$ *and* $\mathrm{ratio_p}$ *are parameters that determine when a process graph is considered to be relevant, potentially relevant or irrelevant and can be set as desired, to produce the best results.*

After determining the sets of process graphs that are relevant and potentially relevant to a query graph, we can rank them. We rank the process graphs in the ‘relevant’ set according to their estimated similarities (GSim) with respect to the query graph. We rank the process graphs in the ‘potentially relevant’ set by computing their similarity to the query graph, using an existing process similarity metric (e.g. one from [6,7]). The complete ranked list is formed by the ranked relevant process models followed by the ranked potentially relevant process models.

5 Evaluation

This section shows how the estimation step affects process similarity search in terms of the quality of the retrieved results and the execution time. It first explains the setup of the evaluation and second the results. We implemented a tool for the technique in this paper (including transferring process models to process graphs automatically).[2]

5.1 Evaluation Setup

We have two experimental setups one for evaluating the quality of retrieved results and one for evaluating the execution time, respectively. We use 10 search models for both of the evaluations, which are derived from the SAP reference model. We made some adaption to these 10 models to better evaluate the technique in this paper, e.g., rephrasing labels (synonyms) and using sub-models.

To evaluate the quality of retrieved results, we use the same evaluation dataset as in [6]. This dataset consists of 100 process models that are derived from the SAP reference model. This is a collection of 604 business process models (described as EPCs) capturing business processes supported by the SAP enterprise system. On average the process models contain 21.6 nodes with a minimum of 3 and a maximum 130 nodes. The average size of node labels is 3.8 words. For each of the 1000 combinations of a search model and a process model a human observer judged whether the process model was relevant to the search model. The R-Precision [4] of the search results returned by a particular similarity search algorithm can then be computed to indicate the quality of those results.

Definition 14 (R-Precision). *Let $\mathcal{D}$ be the set of process models, $\mathcal{Q}$ be the set of query models and* relevant : $\mathcal{Q} \rightarrow \mathbb{P}(\mathcal{D})$ *be the function that returns the set of relevant process models for each query model (as determined by the human observer).*

Given the list of search results $D = [d_1, d_2, ..., d_n]$ for a query q with $\forall d_i \in \mathcal{D}$, the R-Precision is the precision of the first R results, where $R = |\text{relevant}(q)|$ is the total number of process models that is relevant to the query:

$$\text{R-Precision} = \frac{|[d_i \in D | i \leq n, i \leq R, d_i \in \text{relevant}(q)]|}{R}$$

To evaluate the quality of the retrieved results, we compare the R-Precision of the greedy graph similarity search algorithm that we developed in previous work [6] to the R-Precision of the same algorithm with feature-based similarity estimation applied first. We use the greedy graph similarity search algorithm, because it is the fastest algorithm of the ones we studied [6] and, therefore, provides a lower-bound for improvements in execution time.

To evaluate the execution time, we compare the 10 queries with all 604 business process models in the SAP reference model, instead of just the 100 process models. We can do this, because to compute the execution time we do not need the relevance scores.

[2] http://is.tm.tue.nl/research/apromore.html

5.2 Evaluation Results

Table 1 shows the quality of the results that are retrieved by the original greedy similarity search algorithm and those returned by the algorithm in combination with similarity estimation, based on various feature types.

Table 1. Quality of retrieved results

Feature(n)	Occurrences	Matches	Relevant	Potential	Irrelevant	R-Precision
Former Work [6]	-	-	0	100	0	0.84
1:Node(1)	374	581	5.5	10.9	83.6	0.84
2:1+Seq(2)	+267	+197	8.1	8	83.9	0.83
3:2+Seq(3)	+175	+96	7.8	10.1	82.1	0.83
4:2+Split(3)	+87	+93	7.8	10.1	82.1	0.83
5:4+Split(4)	+23	+11	7.8	10.1	82.1	0.83
6:2+Join(3)	+58	+18	7.8	10.1	82.1	0.83
7:6+Join(4)	+14	+1	7.8	10.1	82.1	0.83

The rows in the table show the features that are used to do the feature-based similarity estimation. In the first row no feature-based similarity estimation is done, so this row lists the performance of the original algorithm. In the second row similarity estimation is done based only on node features (of size 1). In the third row similarity estimation is done based on node features plus sequence features of size 2 and so on and so forth (when a feature, whose size is bigger than 1, is evaluated, its parent features are always included).

The columns in the table show the properties of the features and similarity estimation based on the features. First, they show the number of times features of a given type occur in the set of process models and the number of times features of a certain type match in the set of process models. For example, in the set of process models, there could be four nodes labeled 'A'. These nodes count as four occurrences of the node feature type. Because of their high label feature similarity, these nodes can be considered to match. This leads to six matches, because each of the four nodes can be matches to each of the others. Second, the columns show the average number of process models that is estimated as being relevant, potentially relevant and irrelevant over the ten queries. Third, the columns show the average R-Precision over the ten queries.

The table shows that when similarity estimation is done based only on node features on average 5.5 models are estimated to be relevant, 10.9 models to be potentially relevant and 83.6 models as irrelevant. Therefore, in this situation, the original algorithm only has to be used to measure the similarity of about 11% of the total number of process models. In this case the R-Precision remains the same. If sequences of size two are also used to perform the similarity estimation, only 8% of the process models has to be compared using the original algorithm. However, this does lead to a slightly lower R-Precision. Inclusion of other types of features does not improve the similarity estimation any further.

Table 2. Execution time of similarity search

Features(n)	Relevant	Potential	Irrelevant	$T_{estimate}$	$T_{compute}$	T^{avg}_{total}	T^{min}_{total}	T^{max}_{total}
Former Work [6]	0	604	0	0.00s	0.60s	0.60s	0.16s	1.45s
1:Node(1)	7	73	524	0.05s	0.12s	0.17s	0.03s	0.40s
2:1+Seq(2)	13.7	44.9	554.4	0.05s	0.06s	0.11s	0.03s	0.24s
3:2+Seq(3)	9.5	73.2	521.3	0.05s	0.16s	0.21s	0.03s	0.43s
4:2+Split(3)	9.5	73.2	521.3	0.05s	0.16s	0.21s	0.03s	0.43s
5:4+Split(4)	9.5	73.2	521.3	0.05s	0.16s	0.21s	0.03s	0.43s
6:2+Join(3)	9.5	73.2	521.3	0.05s	0.16s	0.21s	0.03s	0.43s
7:6+Join(4)	9.5	73.2	521.3	0.05s	0.16s	0.21s	0.03s	0.43s

Table 2 shows the execution time of the similarity search both when only using the original algorithm and when using certain feature types for similarity estimation. All the experiment are run on a laptop with the Intel Core2 Duo processor T7500 CPU (2.2GHz, 800MHz FSB, 4MB L2 cache), 4 GB DDR2 memory, the Windows Vista operating system and the SUN Java Virtual Machine version 1.6.

The execution time consists of two parts: the total time it takes to estimate the similarity and classify process models as relevant, potentially relevant or irrelevant, denoted $T_{estimate}$; and the time it takes to compute the similarity for the models classified as potentially relevant, denoted $T_{compute}$. Table 2 shows the average estimation and computation times over the ten search queries. In addition to that it shows the average total time over the ten queries and the (minimum) time of processing the query that takes the least time and the (maximum) time of processing the query that takes the most time.

The table shows that if we, on average, estimating similarity based on node features helps to retrieve similar models 3.5 times faster and from table 1 we know that this does not impact the quality of the search results. Also involving the sequence of size two feature type even helps retrieve similar models 5.5 times faster, but from table 1 we know that this reduces the quality of the results by about 1% as a tradeoff.

The table also shows that, on average, the total search time for the original algorithm is quite acceptable and takes only 0.60 seconds. However, in the worst case the total search time for the original algorithm is already 1.45 seconds and this time is linear over the number of models in the collection, meaning that if we were to search a collection of 2000 models (the size of the collection of business process of a large telecom provider in the Netherlands) the search time would already be around 5 seconds in the worst case. This is still much slower than common search engines, e.g., Google.

The technique depends on several parameters:

– dcutoff, which is a parameter that determines whether a role feature is considered to be discriminative (Definition 6).
– $\text{lcutoff}_{\text{high}}$, rcutoff and $\text{lcutoff}_{\text{med}}$, which are parameters that determine what is considered to be a sufficiently similar for a feature to match (Definition 10).

– $\mathrm{ratio_r}$ and $\mathrm{ratio_p}$, which are parameters that determine which class a process model belongs to based on the fraction of features that match with the search query model (Definition 13).

We vary each of these parameters from 0 to 1 in increments of 0.1 and ran the experiments with all possible combinations of parameter values within this range. We use the parameters that, on average, gives the highest R-Precision or the fewest potentially relevant models with respect to the queries to show attractive tradeoffs. The values that we use are $dcutoff = 0.3$, $lcutoff_{high} = 0.8$, $rcutoff = 1.0$ and $lcutoff_{med} = 0.2$. The other two parameters also depend on the type of features we use. For the node feature (the second row in Table 1 or 2), $ratio_r = 0.5$; otherwise, $ratio_r = 0.2$. For the node and sequence (with two nodes) features (the second and third rows in Table 1 or 2), $ratio_p = 0.1$; otherwise, $ratio_p = 0.0$. The values of the parameters are specific to this dataset. More general values for should be obtained by doing more experiments.

6 Related Work

The work presented in this paper is related to business process similarity search techniques and to general graph similarity search techniques, which we use as a basis for the work in this paper.

Business process similarity search techniques have been developed from different angles [1,10,11,12,13,15,19]. These techniques mainly vary with respect to the information, incorporated in the business process models, that they use to determine similarity [6] and the underlying formalism that they use to determine similarity [8]. The work described in this paper complements existing business process similarity search techniques, because it focuses on estimating business process similarity, rather than measuring it exactly, and using that estimate to improve the time performance of existing techniques. As such it can be combined with any of the existing techniques to improve their performance. Lu and Sadiq [12] also use features to determine similarity, but because their goal differs from the goal of this paper (they want to measure similarity exactly), their features are larger than ours, potentially consisting of a complete process model. This makes their features suitable for measuring similarity exactly, but not for estimating it quickly.

General graph similarity search has been applied in various application domains, including fingerprint search, DNA search and chemical compound search. In these domains feature-based methods have been used for similarity estimation and measurement. Willett et al. [18] describe feature-based similarity search in a chemical compound databases. For the structure-based method, ShaSha et al. [17] propose a path-based approach; Yan et al. [20] use discriminative frequent structures to index graphs; Zhao et al. [22] prove that using tree structures and a small number of discriminative graph structures to index graphs is a good choice. Furthermore, Yan et al. [21] also investigate the relationship between feature-based and structure-based methods and built an connection between the two. The main difference between the work that has been done in this area and the work in this paper, is the different nature of business process graphs as compared to graphs in other domains. In particular, there is practically no restriction to the number of possible node labels in a business process graphs and matching nodes do not necessarily have the identical labels. In comparison dna nodes have four possible labels, chemical compound nodes have 117 possible labels, and in both cases matching nodes have identical labels. Also, business process graphs have different structural properties and patterns. These characteristics require that feature types are defined specifically for business process graphs. In addition to that processing feature similarity is different, because business process graphs do not require features to match exactly for graphs to be similar, while graphs in other domains do require features to match exactly.

7 Conclusion

This paper presents a technique for improving the speed of business process similarity search. The evaluation shows that the search time of the fastest algorithm for business process similarity search that exists today can be reduced by a factor 3.5 on average, without impacting the quality of the results. The execution time can even be reduced by a factor 5.5, if a reduction of the quality of the results of 1% is acceptable. These reductions are computed as the average reduction over ten queries. The reduction for the most complex query is a factor 16.5 and the reduction for the least complex query is a factor 2.5.

The technique works by quickly classifying models in a collection as either relevant, irrelevant or potentially relevant to a search query. The original algorithm then only has to be used to classify the potentially relevant models in the collection. The classification is done based on simple, but representative, parts of business process models, also called features. The evaluation shows that the factor 3.5 reduction of processing time is achieved, if the labels and the interconnections of individual nodes from a business process model are used as features to estimate the similarity. The factor 5.5 reduction of processing time with a reduction of 1% on the quality of the search results is achieved, if individual nodes, their interconnections and sequences of two nodes are used as features to estimate the similarity. Other features that have been used are sequences of three nodes and splits and joins. However, these features do not further improve the quality of the search results or reduce the search time.

Let k be the number of process models in the collection and n be the maximum number of nodes in a single process model. To determine similarity of process models based on node features, for each node in the search graph, similar nodes need to be selected from all the nodes in the M-Tree. There are at most $k \cdot n$ nodes in the M-Tree when all the nodes are distinct from each other. Therefore, the time complexity of node feature similarity has an upper bound of $O(n \cdot log(k \cdot n))$, while the time complexity of the, currently fastest, greedy algorithm for process similarity search is $O(k \cdot n^3)$ [6]. The complexity of similarity with respect to other features is linear, because it depends on node feature similarity of which the results can be stored.

There are some drawbacks to the technique in this paper. First, the estimation is mainly based on label similarity. However, similar tasks can be labeled differently, e.g., synonyms, different levels of verbosity. The paper uses discriminative role features to deal with this issue, which have proven to be effective in the context of our evaluation data set. However, in our evaluation data set process models were constructed by the same people, leading to models that have similar labels for similar tasks. This may not always be the case. Therefore, we applied more advanced metrics for label similarity that consider synonyms [7] and domain ontologies [10]. The integration of these advanced metrics into the technique described in this paper is left for future work. Second, the technique in this paper mainly focuses on tasks and connections between them. However, process models often contain more information that may be exploited when determining their similarity, e.g., resources and data used. Using this information when determining process similarity is left for future work.

Acknowledgement

The research is supported by the China Scholarship Council (CSC).

References

1. van der Aalst, W.M.P., de Medeiros, A.K.A., Weijters, A.J.M.M.T.: Process Equivalence: Comparing Two Process Models based on Observed Behavior. In: Dustdar, S., Fiadeiro, J.L., Sheth, A.P. (eds.) BPM 2006. LNCS, vol. 4102, pp. 129–144. Springer, Heidelberg (2006)
2. Awad, A.: BPMN-Q: A language to query business processes. In: Proceedings of EMISA 2007, Nanjing, China, pp. 115–128 (2007)
3. Bartolini, I., Ciaccia, P., Patella, M.: String Matching with Metric Trees Using an Approximate Distance. In: Laender, A.H.F., Oliveira, A.L. (eds.) SPIRE 2002. LNCS, vol. 2476, p. 271. Springer, Heidelberg (2002)
4. Buckley, C., Voorhees, E.M.: Evaluating Evaluation Measure Stability. In: Proceedings of the ACM SIGIR Conference, pp. 33–40 (2000)
5. Choi, I., Kim, K., Jang, M.: An xml-based process repository and process query language for integrated process management. Knowledge and Process Management 14(4), 303–316 (2007)
6. Dijkman, R.M., Dumas, M., García-Bañuelos, L.: Graph Matching Algorithms for Business Process Model Similarity Search. In: Dayal, U., Eder, J., Koehler, J., Reijers, H.A. (eds.) Business Process Management. LNCS, vol. 5701, pp. 48–63. Springer, Heidelberg (2009)
7. van Dongen, B.F., Dijkman, R.M., Mendling, J.: Measuring Similarity between Business Process Models. In: Bellahsène, Z., Léonard, M. (eds.) CAiSE 2008. LNCS, vol. 5074, pp. 450–464. Springer, Heidelberg (2008)
8. Dumas, M., García-Bañuelos, L., Dijkman, R.M.: Similarity Search of Business Process Models. Technical Committee on Data Engineering 32(3), 23–28 (2009)
9. Documentair structuurplan, `http://www.model-dsp.nl`
10. Ehrig, M., Koschmider, A., Oberweis, A.: Measuring similarity between semantic business process models. In: Proceedings of the 4th Asia-Pacific Conference on Conceptual Modelling, Ballarat, Victoria, Australia, pp. 71–80 (2007)

11. Li, C., Reichert, M.U., Wombacher, A.: On Measuring Process Model Similarity based on High-level Change Operations. In: Li, Q., Spaccapietra, S., Yu, E., Olivé, A. (eds.) ER 2008. LNCS, vol. 5231, pp. 248–264. Springer, Heidelberg (2008)
12. Lu, R., Sadiq, S.K.: On the Discovery of Preferred Work Practice through Business Process Variants. In: Parent, C., Schewe, K.-D., Storey, V.C., Thalheim, B. (eds.) ER 2007. LNCS, vol. 4801, pp. 165–180. Springer, Heidelberg (2007)
13. Madhusudan, T., Zhao, L., Marshall, B.: A Case-based Reasoning Framework for Workflow Model Management. Data Knowledge Engineering 50(1), 87–115 (2004)
14. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press, Cambridge (2008)
15. Minor, M., Tartakovski, A., Bergmann, R.: Representation and structure-based similarity assessment for agile workflows. In: Weber, R.O., Richter, M.M. (eds.) ICCBR 2007. LNCS (LNAI), vol. 4626, pp. 224–238. Springer, Heidelberg (2007)
16. Curran, T.A., Keller, G.: SAP R/3 Business Blueprint - Business Engineering mit den R/3-Referenzprozessen. Addison-Wesley, Bonn (1999)
17. ShaSha, D., Wang, J., Giugno, R.: Algorithmics and applications of tree and graph searching. In: Proceedings of the 21th ACM International Symposium on Principles of Database Systems, pp. 39–53 (2002)
18. Willett, P., Barnard, J., Downs, G.: Chemical similarity searching. J. Chem. Inf. Comput. Sci. 38, 983–996 (1998)
19. Wombacher, A.: Evaluation of technical measures for workflow similarity based on a pilot study. In: Meersman, R., Tari, Z. (eds.) OTM 2006. LNCS, vol. 4275, pp. 255–272. Springer, Heidelberg (2006)
20. Yan, X., Yu, P.S., Han, J.: Graph Indexing: A Frequent Structure-based Approach. In: Proceedings of the 2004 ACM SIGMOD, pp. 335–346 (2004)
21. Yan, X., Yu, P.S., Han, J.: Substructure Similarity Search in Graph Databases. In: Proceedings of the 2005 ACM SIGMOD, pp. 766–777 (2005)
22. Zhao, P., Yu, J.X., Yu, P.S.: Graph Indexing: Tree + Delta >= Graph. In: Proceedings of the 2007 ACM VLDB, pp. 938–949 (2007)