Flexible Process Model Mapping using Relaxation Labeling

Luis Delicado, Josep Carmona, and Lluís Padró

Computer Science Department
Universitat Politècnica de Catalunya
Barcelona, Spain
{ldelicado,jcarmona,padro}@cs.upc.edu

Abstract. Computing a mapping between two process models is a crucial technique, since it enables reasoning and operating across processes, like providing a similarity score between two processes, or merging different process variants to generate a consolidated process model. In this paper we present a new flexible technique for process model mapping, based on the relaxation labeling constraint satisfaction algorithm. The technique can be instantiated so that different modes are devised, depending on the context. For instance, it can be adapted to the case where one of the mapped process models is incomplete, or it can be used to ground an adaptable similarity measure between process models. The approach has been implemented inside the open platform `NLP4BPM`, providing a visualization of the performed mappings and computed similarity scores. The experimental results witness the flexibility and usefulness of the technique proposed.

1 Introduction

Process models are crucial elements for representing in a precise way the real processes of an organization. Their use goes beyond the mere modeling purpose: on the one hand, *BPMS* platforms, which allow designing, deploying and managing the processes in organizations, are based on process models. On the other hand, evidence-based process models (i.e., process models with a high alignment with respect to the underlying real process) can be used to analyze the process formally, e.g., detecting inconsistencies or performance problems that may hamper the correct and optimal execution of the process. Furthermore, the existence of environments for creating, managing and querying *process model collections* enable a high-level analysis of process models.

Hence it is of paramount importance to automate the comparison of process models. Due to its importance, this problem has received significant attention in the BPM field; contributions can be split into *structural techniques* based on graph-edit distance [1–4], and *behavioural techniques* that focus on the execution semantics or behavioural relations of the corresponding models [5–9]. One key enabler of process model comparison is the generation of a mapping between the elements of the two process models [10]. The available techniques for automated

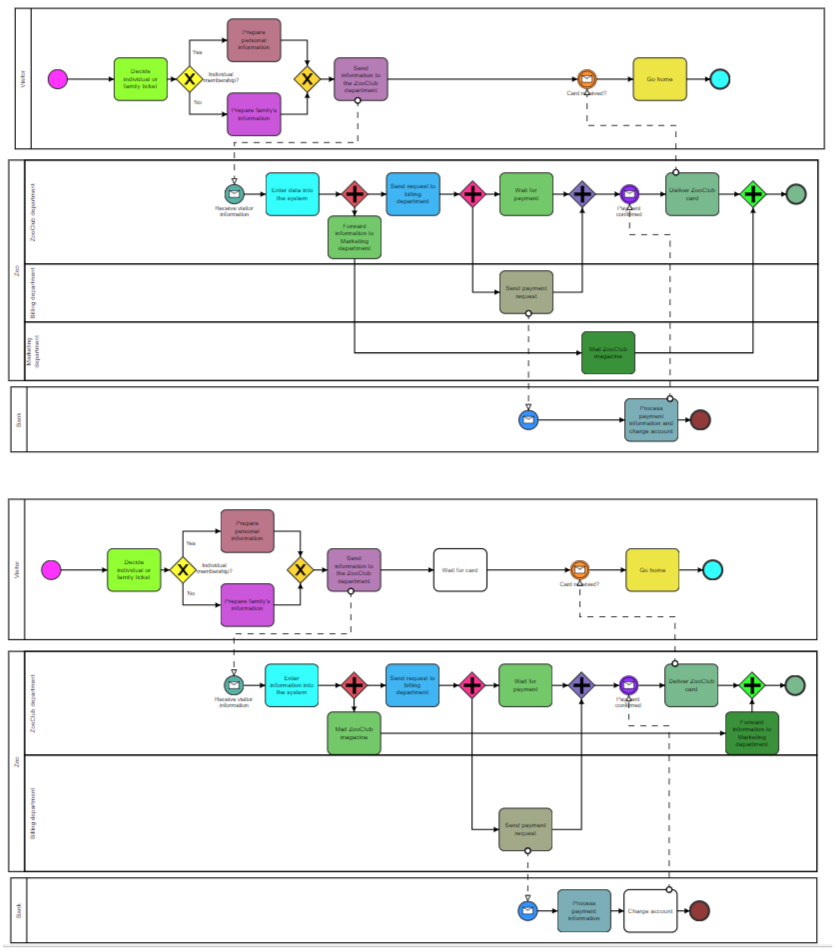
Visitor
Decide individual or family ticket
Individual membership?
Yes
No
Prepare personal information
Prepare family's information
Send information to the ZooClub department
Card received?
Go home
Zoo
ZooClub department
Receive visitor information
Enter data into the system
Send request to billing department
Wait for payment
Payment confirmed
Deliver ZooClub card
Forward information to Marketing department
Billing department
Send payment request
Marketing department
Mail ZooClub magazine
Bank
Process payment information and charge account
Visitor
Decide individual or family ticket
Individual membership?
Yes
No
Prepare personal information
Prepare family's information
Send information to the ZooClub department
Wait for card
Card received?
Go home
Zoo
ZooClub department
Receive visitor information
Enter information into the system
Send request to billing department
Wait for payment
Payment confirmed
Deliver ZooClub card
Mail ZooClub magazine
Forward information to Marketing department
Billing department
Send payment request
Bank
Process payment information
Charge account

Fig. 1. Visualization of the technique of this paper in the platform `NLP4BPM`.

process model mapping are sometimes unable to capture different context of application, like for instance when process models are incomplete, or when the emphasis in the mapping needs to change depending on the application scenario.

As reported in a recent survey on process model similarity [10], there is a need for an analysis of different usage scenarios of similarity metrics:

> [10] *"... it is necessary to analyze in how far particular similarity measures match the requirements of specific usage scenarios and whether it is meaningful to apply different similarity measures to different scenarios."*

In this paper we propose a novel method to compare process models in a flexible way, so that the context of application can be taken into account by selecting a particular mode when mapping the elements of the two processes. The intuitive idea is to encode the problem as a *Relaxation Labeling* (RL) optimization problem, so that an optimal assignment between the elements of the two process models is computed, according to a configurable set of constraints. The used constraints can focus on structure, labels, semantics, etc., and be adapted to each particular problem. Thus, semantics of the model are also taken into account by (i) performing a full linguistic analysis of the text associated to the different process model elements, (ii) considering also the mapping of the routing elements such as gateways and its connections, and (iii) consider the process model lanes. Several examples of operation modes are envisioned:

– A general mode, where both the structure and the text are considered.
– A structural mode, where the structure is dominant in the comparison.
– A text-based mode, where the linguistic analysis is dominant in the comparison.
– A partial mode, where one of the process models may be incomplete.

In particular, the case where one of the two process models is incomplete can be used in e-learning context, by providing students continuous feedback on their progress in a process modeling course. At the end of the paper, we will report preliminary experiments in this context.

The approach has been incorporated in the `NLP4BPM` open platform [1], which provides a friendly visualization of the optimal mapping found. Figure 1 shows an example of a mapping found.

2 General Approach: From Mapping to Similarity Score

To contextualize the different contributions of this paper, in this section we provide a general perspective on the necessary steps followed to assess the similarity of two process models.

We assume process models to be specified in BPMN 2.0 notation, which is a widespread notation for modeling processes in organizations. Section 3.1 provides the main constructs considered in this paper.

The first step is to perform a complete linguistic analysis of the text in the two process models. This is of paramount importance, since it enables a comparison that goes significantly beyond a mere string distance of the set of original labels. Section 3.2 provides a detailed description of this step.

Once the linguistic analysis is performed, the mapping between the elements of the two process models is computed. The computation of this mapping is carried out using a RL algorithm, a technique that is introduced in Section 3.3. The encoding of the mapping problem as a relaxation labeling instance is the main contribution of this paper, and is described in Section 4. In Section 4.4 it is shown how to adapt the encoding so that incomplete process models are considered in the mapping, an scenario that can be found in the educational context.

[1] http://nlp4bpm.cs.upc.edu

When a mapping is obtained, a similarity measure can be computed. The metric proposed in this paper relies in mapping one model to the other in both directions, and then taking into account the number of bidirectional matches, the total number of matches, and the matches of the neighbors of the bidirectional matches. See details in Section 5.

3 Preliminaries

3.1 Process Models

There are a large variety of graphical notations to represent model processes. In this paper we focus on BPMN 2.0, notation created as a standard for business process modeling. However, the techniques explained in this paper can be used with other notations like EPCs, Petri Nets or others.

BPMN has three different kinds of elements. First, the main elements are the nodes in the diagram, which may belong to three different types: *Events*, which represent that something happens; *Activities*, which represents some task that is performed; and *Gateways*, which split or join flow control. Second, the notation has different edges to connect nodes. A solid line indicates the process workflow, while dashed lines represent messages sent between process participants. Finally, there are organization elements such as *lanes* that contain activities performed by the same participant, and *pools*, that group several related lanes. Figure 1 shows an example of a BPMN model.

3.2 Linguistic Analysis of the Textual Information

In order to better align tasks, we do not rely just on the words occurring in the task labels, but also on their semantics. For that we perform a full NLP analysis on the tasks labels, obtaining a semantic representation of them. Particularly, we focus on the *action* being performed, the *agent* who performs the action and the *patient* upon which the action is performed. Additionally, we also extract per-word information such as the part of speech (PoS) tag and its ontological sense in WordNet [11]. By collecting this lower-level information, we allow our technique to fall back to the word level whenever the predicate level description is ambiguous or incomplete.

Thus, we process the natural language bits contained on the model labels (from activities, events, gateways, arcs, lanes and pools). However, these fragments are not whole sentences but isolated phrases or chunks, which causes the standard NLP tools to suffer a loss in performance. Moreover, the model structure itself provides some clues about the information to extract from each element (e.g. lane labels are typically noun phrases describing the agent of the tasks in the lane, while task labels describe the action and the patient in usually simple patterns).

In order to properly extract the required information, we followed the idea of [12, 13] and devised an ad-hoc NLP analysis pipeline, based on the customizable FreeLing NLP library[2] [14].

– General purpose tokenization, morphological analysis, and word sense disambiguation modules are used.
– The general purpose PoS tagger is used, but configured to provide the k-best solutions instead of just the most likely one. This will compensate for some errors due to analyzing chunks and not whole sentences. For instance, for the task label "mail response", the word "mail" is wrongly tagged as a noun by the generic tagger, because complete English sentences seldom start with a verb. If we get more than one solution from the tagger, we will find that the second-best answer tags "mail" as a verb.
– We use a rule-based parser with an ad-hoc CFG grammar to detect patterns and phrases inside the text fragments. All k-best tag sequences produced by the tagger are fed to the grammar, and the first one matching one valid pattern is used.
  Patterns covered so far by this grammar are: verb actions (e.g. *send report*), noun actions (e.g. *data transmission*), and conjunctions of them (e.g. *send report and notify user*). Verb actions can have the following patterns, where the verb is extracted as the *action*, the noun phrase as the *patient*, and the prepositional phrase as *complement*:
  • verb (e.g. *request*, *check*, *reject*)
  • verb + noun-phrase (e.g. *store physical file*, *Create job description*)
  • verb + prep-phrase (e.g. *Ask for details*, *Inform about the procedure*)
  • verb + noun-phrase + prep-phrase (e.g. *Ask user for details*, *Inform patient about the risk of the procedure*)
  Noun actions can have the following patterns:
  • noun-phrase (e.g. *receipt payment*, *master data transmission*)
  • noun-phrase + prep-phrase (e.g. *reception of packaged goods*, *data collection for billing adjustment*)
  Additionally, the grammar detects the head word of each component, so we can extract that *file* is the main word in the noun phrase *store physical file*, *risk* is the head of *the risk of the procedure*, and *transmission* is the head of *master data transmission.*

3.3 Relaxation Labeling

Relaxation labeling (RL) is a family of iterative algorithms that perform cost function optimization over a set of variables that must be assigned appropriate discrete values (or *labels*). The algorithm requires a set of soft constraints that state the (in)compatibility of certain variable-value combinations, and computes a *consistent labeling*, i.e. a set of assignments that best satisfies the constraints.

[2] `http://nlp.cs.upc.edu/freeling`

RL algorithms are approximate optimizers that find local optima and are closely related to gradient step, simulated annealing, and neural network optimization [15]. They were originally devised in computer vision field, but have been applied to several other AI tasks, graph matching among them ([16]).

Interesting properties of the algorithm are that each variable can have a different set of possible labels, and that any kind of constraint relating combinations of any number of pairs (variable,label) can be used, thus providing a very flexible setting that can be used to heuristically solve many combinatorial-space search problems.

Advantages of RL for graph matching over other approaches –such as graph edit distance– include: the generality of the approach (any problem stated in terms of variables, labels, and constraints can be solved by the same algorithm) and the problem description flexibility (constraints can encode many kinds of restrictions and take into account different types of information).

4 Flexible Mapping Through Relaxation Labeling

In this section we describe the relaxation labeling algorithm and how we apply it to the graph matching problem at hand.

4.1 Modeling Graph Mapping as a Relaxation Labeling Problem

To model a task into a RL problem, we must establish which components are variables, which are the possible labels for each variable, and which are the constraints favoring or penalizing certain variable-label assignment combinations. In our case, we want to map one graph into another. Using RL for graph matching has been proven successful in the past [16], and we follow a similar approach.

We will consider one of the process models to map as the *source* model and the other as the *target* model. Each model element (task, gateway, event, ...) in the source model will correspond to a RL variable. The possible values (or labels) for this variable will be each element in the target model. Given this construction, each source model element will be assigned exactly one target model element.

The problem representation also requires constraints that penalize wrong assignments, and favor appropriate pairings. For instance, we can add constraints to penalize assignments violating flow control order, or to reward assignments that respect it. E.g. if source model element s_1 precedes source model element s_2 and target element t_1 precedes target element t_2, then a solution containing both pairs (s_1, t_1) and (s_2, t_2) must be rewarded, but if t_1 follows t_2, the combination must be penalized.

4.2 Assignment Initialization

Relaxation labelling performs continuous optimization on a probabilistic representation (i.e. each variable has a probability distribution over the set of its possible values) that is updated iteratively until convergence. Since it performs a local search, choosing a good starting point reduces the number of required iterations, and may help to find better solutions if the cost function is not convex.

We implemented three different initialization strategies: random, uniform, and informed. Informed initialization uses source and target tasks features, such as their text labels, number of connections, etc. to establish the initial assignment values. For this, we use FreeLing NLP tools to extract linguistic information from the task labels, such as the lemmas, parts-of-speech, or WordNet synsets, as described in section 3.2.

4.3 Constraints

The Relaxation Labeling algorithm uses a set of constrains that determine a weight for all matches between elements of the two models. Our model uses 8 constraint patterns that are instantiated for each variable. Constraints state (in)compatibility of a label assignation for one variable with the assignation of its neighbors (e.g. the combination of source element s_1 being assigned to target t_1 and source s_2 being assigned to target t_2, is compatible if t_1 and t_2 have the same neighboring relation than s_1 and s_2).

Constraints are not binary, but they express a *degree* of compatibility (which may be negative to express incompatibility). The algorithm will favor combinations with highest total compatibility and penalize those with lower scores.

We use four constraint templates that use only information on the source and target elements, but not on their neighbors:

- **Type:** The assignment of value t_i to variable s_i is compatible if both elements have the same type (e.g. manual task, service, gateway, event, etc.).
- **Name:** The assignment of value t_i to variable s_i is compatible if both elements share linguistic features (e.g. same verbs/actions, same objects, same concepts, ...). FreeLing NLP analyzers are used to extract linguistic features from the element description texts. Obtained feature sets are compared using a similarity function [17] that will determine the degree of compatibility of the two elements.
- **Lane:** The assignment of value t_i to variable s_i is compatible if the names of their lanes are equal.
- **Number of input (output) edges:** The assignment of value t_i to variable s_i is compatible if they have the same number of input (output) edges.
- **Number of input (output) messages:** The assignment of value t_i to variable s_i is compatible if they receive (send) messages and have the same number of input (output) messages.

A second group of constraints takes into account the assignments of neighboring elements[3]:

[3] Note that in the constraints we talk about elements (tasks, gates, events), not only BPMN tasks. For the sake of understandability, in the figures we illustrate the neighboring constraints over tasks.

– **Neighbor 1:** (Figure 2) The assignment of value t_i to variable s_i is compatible if the element s_i has an immediately following (preceding) element s_j that is assigned a value t_j that also immediately follows (precedes) t_i. The degree of compatibility is a fixed value, given as a parameter to the algorithm.

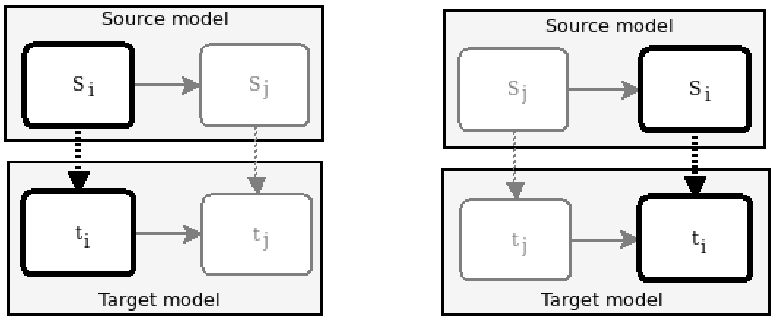

Fig. 2. Following (left) and preceding (right) neighbour 1 constraint

– **Neighbor 2:** (Figure 3) The assignment of value t_i to variable s_i is compatible if the element s_i has two immediately following (preceding) consecutive element s_j, s_k that are respectively assigned values t_j, t_k which are consecutive and also immediately follow (precede) t_j. The degree of compatibility is a fixed value, given as a parameter to the algorithm.

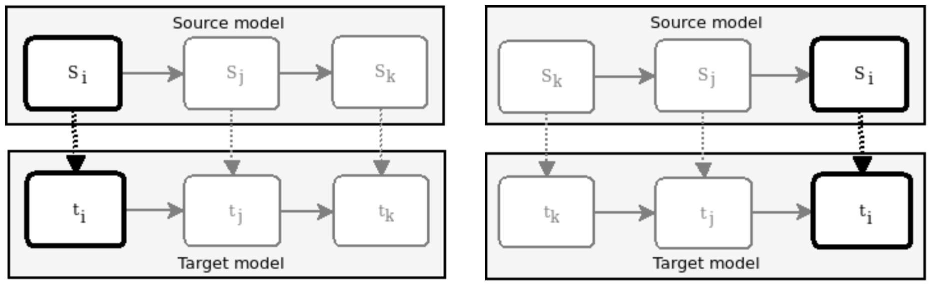

Fig. 3. Following (left) and preceding (right) neighbour 2 constraint

– **Neighbor skip:** (Figure 4) The assignment of value t_i to variable s_i is compatible if the element s_i has an immediately following (preceding) element s_j that is assigned a value t_k that follows (precedes) the element immediately following (preceding) t_i. This constraint also applies if value t_i has an immediately following (preceding) element t_j that is assigned to element s_k that follows (precedes) the element immediately following (preceding) s_i.

The degree of compatibility is a fixed value, given as a parameter to the algorithm. This constraint is intended to capture cases where some of the tasks have been fused together in one of the models, and will usually have a lower compatibility score than the constraints presented above.

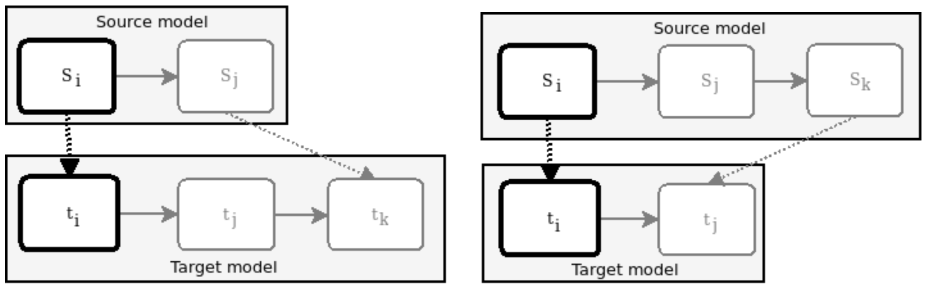

Fig. 4. Both versions of following neighbour-skip constraint. Preceding constraints are symmetrical to these.

– **Inverted neighbor:** (Figure 5) The assignment of value t_j to variable s_j is *incompatible* if the element s_s has an immediately following (preceding) element s_k that is assigned a value t_i that also immediately follows (precedes) t_j. The degree of compatibility is a *negative* fixed value, given as a parameter to the algorithm.

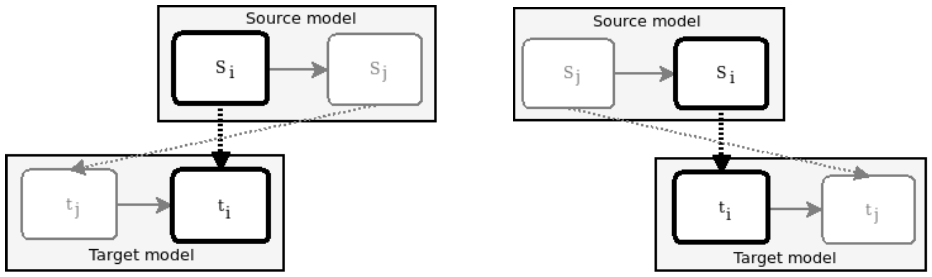

Fig. 5. Following (left) and preceding (right) inverted neighbour constraint

– **Neighbor messages:** This constraint is similar to *Neighbor 1* but using message connections. The assignment of value t_i to variable s_i is compatible if the element s_i sends (receives) a message from element s_j that is assigned a value t_j that also sends (receives) a message from t_i. The degree of compatibility is a fixed value, given as a parameter to the algorithm.

When the algorithm converges, each variable will –typically– have all its probability mass in a single label, thus establishing a unique-label-per-variable mapping. However, in some cases (e.g. if constraints are too vague or contradict each other) some variables can end with their probability mass evenly distributed among more than one value. For these cases, a threshold is used to determine whether the mapping is included in the final solution.

4.4 Mapping Incomplete Process Models

The presented method can be adapted to map the elements of an incomplete model to a complete model. One possible scenario for this task is the an educational setting, where a process modeling student is incrementally completing a model (see Section 6.3). Computing a mapping between this partial model and a gold solution (provided by the teacher) can provide valuable feedback to the student, not only when the modeling task is finished and delivered, but also during the different steps followed to accomplish the modeling task. This feedback can be integrated into a model editor that guides the student through the task.

Mapping a partial model poses some challenges to the approach presented above: In an incomplete model, there will be missing tasks and edges, and their absence should not be penalized, since we want to assess the correctness of the model built so far, not of the parts not addressed by the student yet.

The algorithm described in previous sections tries to match every source model task to a target model task, in both directions. When mapping from a partial (thus smaller) model to a gold complete model, presumably all tasks will find a correspondence. But when performing the reverse mapping, there will be tasks in the gold model that do not exist in the partial model yet. Thus, we add a dummy task in the partial model without any label nor any connection. We also add a very low-weight constraint that allows any gold model task to be mapped to the dummy task. This will result in any gold model task being mapped to the dummy task only if there is not any other constraint with a higher weight directing it somewhere else. This idea is illustrated in Figure 6, where the student modeled the initial and final steps of the process, but has not finished the intermediate parts yet. The gold model elements missing in the partial model are matched with the dummy task because there is not any evidence for an alternative decision.

5 Using the Mapping to Compute Similarity Metrics

The mapping between processes built using the algorithm described above can be used to compute similarity measure between both input models.

For this, we apply the mapping process twice, one in each direction (i.e. use tasks in model 1 as variables and tasks in model 2 as values, and viceversa), and then we compute

$$sim = \lambda \frac{2 * |A \cap B|}{|A| + |B|} + (1 - \lambda) \frac{C_{eq}}{C_{eq} + C_{neq}}$$

Where,

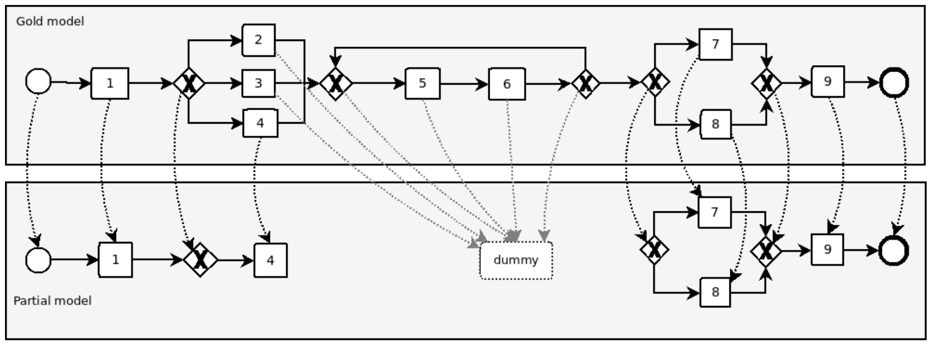

Fig. 6. Dummy element to match a non-connected partial model

- A and B are the sets of task pairings produced by each run of the algorithm, and $A \cap B$ is the set of task pairings present in both mappings.
- C_{eq} is the number of pairs $\{(t_i, s_i), (t_j, s_j)\} \in (A \cap B) \times (A \cap B)$ such that t_i follows (precedes) t_j and s_i follows (precedes) s_j
- C_{neq} is the number of pairs $\{(t_i, s_i), (t_j, s_j)\} \in (A \cap B) \times (A \cap B)$ such that t_i follows (precedes) t_j but s_i does not follow (precede) s_j, or viceversa.

Thus, the similarity measure is a weighted average between Sorensen-Dice similarity between the sets A and B and the percentage of connections that are coincident in both mappings. The first term evaluates the coincidence of task mappings in each direction, and is intended to capture the similarity of the tasks in both models. The second term tries to measure to which extent their connections are alike.

5.1 Similarity of Partial Models

Computing the similarity of a partial model with a complete model (e.g. for the educational scenario described above) requires some adaptation of the similarity measure, in order to avoid penalizing the partial model for the missing elements.

Thus, for the case of partial model matching, we add the dummy task as described in Section 4.4, and do not count (neither as errors nor hits) the pairings of any gold task with the dummy. In consequence, the similarity metric used for partial model mapping is the same than in the general case, though it is applied to different pairing sets:

$$sim = \lambda \frac{2 * |A \cap B|}{|A| + |B|} + (1 - \lambda) \frac{C_{eq}}{C_{eq} + C_{neq}}$$

Where,

- A and B are the sets of task pairings produced by each run of the algorithm *without pairings with dummy elements*, and $A \cap B$ is the set of task pairings present in both mappings.

– C_{eq} and C_{eq} are the same as before, though computed on the A and B sets without pairs involving dummy elements.

However, it might be that the student simply forgot or skipped the middle tasks and thus, deserves a lower score. So, we need to take into account whether the student actually missed a model element, or there is still chance he/she is going to add it. This can be detected checking the edges between the mapped elements:

If two gold model elements g_1 and g_2 are mapped to partial model elements p_1 and p_2 respectively, and there is no edge between g_1 and g_2, we check for edges between p_1 and p_2:

– If there is no edge connecting p_1 and p_2, then we assume that there is still a chance that the student will add the missing elements in the gap. Thus, we add the dummy as a possible option for any element between g_1 and g_2, so they are excluded from sets A and B and do not penalize the score.
– If there is an edge directly connecting p_1 and p_2, then we assume that the student missed some intermediate elements. Thus, the dummy element is not added as a possible option for the elements between g_1 and g_2, so the missing matches will lower the score.

6 Experiments and Results

In this section we present experiments on both complete and partial model matching scenarios.

6.1 Comparative With Other Metrics

Our first experiment consists of comparing our proposed metric –for complete models– to other metrics in the literature. For that, we rely on the thorough analysis performed in [18] where a gold model (named V_0) is compared with seven variants with increasing divergences (V_1, ... V_7) using a wide range of existing metrics.

Our metric depends on the results of the RL mapping, which has different parameters. The metric itself also has the parameter λ that balances the weight of element matching and connection matching.

We tested our metric under different parameterizations. On the one hand, we tested different values of λ (low lambda values give more weight to model structure and edges, while high values rely more on label similarities) applied on two different settings for the mapping RL algorithm: (a) a setting where structural constraints have higher weight, and thus the mapping is mostly driven by connections and control flow (CF), and (b) a setting where mapping is more influenced by label similarity constraints (LS) –which are trivial in this case since labels are numeric.

The combination of these parameters produce a variety of mappings, and consequently, of similarity scores. We tested both RL settings (CF and LS) combined with three values of λ (0.2, 0.5, 0.8). To compare our metric with the state of the art metrics reviewed in [18], we took the vector (V_1, ... V_7) produced by each metric as a description of its behavior (in the scenario proposed in that survey). Then, we computed the vector for each configuration of our measure, obtaining a cloud of points in a 7-dimension space, each point representing a different metric.

To visualize the relative position of our metric with respect to the metrics reported in [18], we performed a PCA projection to 2D of the points, shown in Figure 7. PCA projects the 7-dimension vectors to the 2D plane that best preserves the distances between the points. Red squares correspond to metrics in the original survey (numbered in the order they appear in the table). Blue triangles correspond to different configurations of our metric.

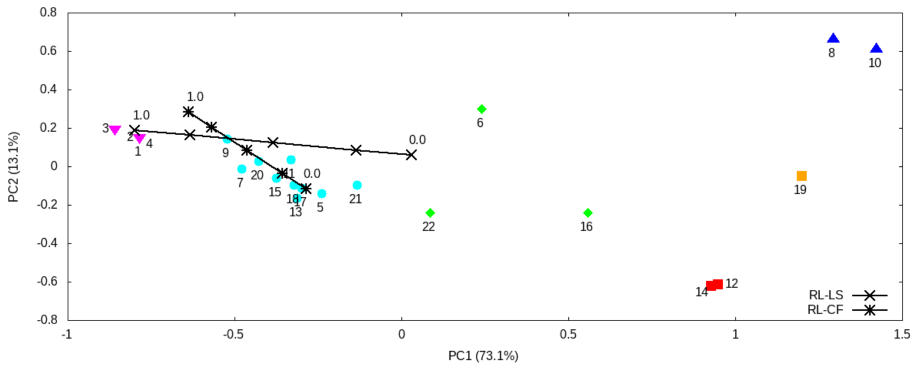

Fig. 7. PCA projection to 2D of the vectors for metrics in [18] (red squares) and different configurations of our metric (blue triangles). The figure shows the 2D projection of the 7-dimension vectors that best preserves the distances between the points.

We can observe how our metric can be tuned to capture different aspects of the compared models. Parameter λ can be used to tune the distance to the center of the cloud (the higher λ is, the closer to the center) while changing the underlying mapping shifts the metric to different regions of the space. Although the interpretation of the effect of this parameters is not obvious and requires further analysis, our approach is promising with regard to offering a general model to approximate a variety of existing metrics.

6.2 Process Model Matching Contest 2015

As a second experiment, we executed our algorithms on some models used in the Process Model Matching Contest 2015 [19]. This dataset consists of 36 pairwise mappings on 9 German universities Master student application process. For each pair, the dataset provides a gold match (sometimes partial). Table 1 is extracted from [19] and presents the results of participating systems. The highlighted row shows the results of our approach, after optimizing the parameters for this dataset. Parameter optimization was performed using a genetic algorithm to find a good local optima in the parameter space.

Table 1. Process Model Matching Contest 2015 results.

	Precision			Recall			F-Measure		
Approach	∅-mic	∅-mac	SD	∅-mic	∅-mac	SD	∅-mic	∅-mac	SD
RMM/NHCM	.686	.597	.248	.651	.610	.277	**.668**	.566	.224
RMM/NLM	.768	.673	.261	.543	.466	.279	.636	.509	.236
MSSS	**.807**	**.855**	.232	.487	.343	.353	.608	.378	.343
OPBOT	.598	.636	.335	.603	.623	.312	.601	**.603**	.300
RL	.523	.450	.318	.577	.495	.309	.550	.472	.285
KMSSS	.513	.386	.320	.578	.402	.357	.544	.374	.305
RMM/SMSL	.511	.445	.239	.578	.578	.336	.543	.477	.253
TripleS	.487	.685	.329	.483	.297	.361	.485	.249	.278
BPLangMatch	.365	.291	.229	.435	.314	.265	.397	.295	.236
KnoMa-Proc	.337	.223	.282	.474	.292	.329	.394	.243	.285
AML-PM	.269	.250	.205	**.672**	**.626**	.319	.385	.341	.236
RMM/VM2	.214	.186	.227	.466	.332	.283	.293	.227	.246
pPalm-DS	.162	.125	.157	.578	.381	.38	.253	.180	.209

Results show that our metric ranks is in the top 5 of the participants, despite the fact that this particular contest scenario characteristics make it harder for a complete mapping approach like ours: Our method matches two task if they are similar enough in description and control flow, but the contest gold standard is very restrictive and some tasks pairs are not annotated despite having the same behavior in the process. For example, in some cases the gold standard matches two task but not the tasks preceeding each of them, even they describe a similar action. Since our algorithm will match the preceeding tasks too, this will be scored as a false positive.

6.3 Experiments on Partial Models

To evaluate the performance of the similarity metric when applied to partial models, we used a dataset coming from a process modeling course in TU/e (Eindhoven), that captures the evolution of a process model while it is being created by different process modeling students. We computed the similarity metric at each intermediate step for a sample of 6 different students. The evolution of the metric while the students refine their models is shown in Figure 8.

It is interesting to note that there are different behavioral patterns that correspond to different student strategies to build the model: Some lines are clearly ascending, indicating that the student does not commit much errors, and steadily completes his model, while others are more erratic, probably due to changes of mind or corrections by the modeler. The study of this dataset and

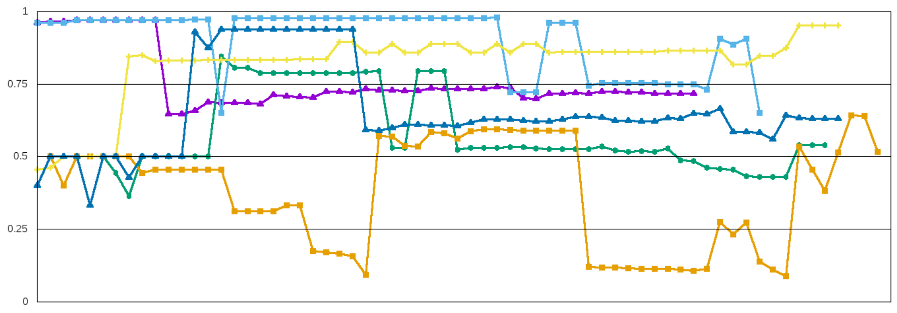

Fig. 8. Similarity metric evolution during the modeling process by different students.

the derived behavior patterns opens a promising line of research to be pursued in the future.

7 Conclusions and Future Work

In this paper we have proposed a new flexible approach for aligning BPMN models. The technique can be instantiated towards very different usage scenarios, and can even deal with partial process models. We also proposed a similarity metric based on these alignments, also adapted to the case of partial models.

We foresee many directions to follow as future work. First, we will consider solving the two mappings as a single RL problem, so that the exploration for solutions can consider jointly the decisions of each direction in order to find better solutions. Also, the derivation of a set of modes that can cover different usage scenarios may be studied, with special interest in partial model mapping for educational applications. Finally, exploring different strategies to compute threshold values so that a better characterization of unmapped tasks will be considered in the near future.

Acknowledgements

We are grateful to Andrea Burattin, Barbara Weber and Hajo Reijers for sharing with us part of the datasets used in this paper. This work has been supported by MINECO and FEDER funds under grant TIN2017-86727-C2-1-R.

References

1. Dijkman, R.M., Dumas, M., García-Bañuelos, L.: Graph matching algorithms for business process model similarity search. In: BPM 2009, Ulm, Germany, September 8-10. (2009) 48–63

2. Dijkman, R.M., Dumas, M., García-Bañuelos, L., Käärik, R.: Aligning business process models. In: EDOC 2009, 1-4 September 2009, Auckland, New Zealand. (2009) 45–53
3. Dijkman, R.M., Dumas, M., van Dongen, B.F., Käärik, R., Mendling, J.: Similarity of business process models: Metrics and evaluation. Inf. Syst. **36**(2) (2011) 498–516
4. Yan, Z., Dijkman, R.M., Grefen, P.W.P.J.: Fast business process similarity search. Distributed and Parallel Databases **30**(2) (2012) 105–144
5. Armas-Cervantes, A., Baldan, P., Marlon, D., García-Bañuelos, L.: Behavioral comparison of process models based on canonically reduced event structures. Business Process Management: 12th International Conference, BPM 2014, Haifa, Israel, September 7-11, 2014. Proceedings (2014) 267–282
6. Dijkman, R.M.: Diagnosing differences between business process models. In: BPM 2008, Milan, Italy, September 2-4. (2008) 261–277
7. Polyvyanyy, A., Weidlich, M., Conforti, R., Rosa, M.L., ter Hofstede, A.H.M.: The 4c spectrum of fundamental behavioral relations for concurrent systems. In: PETRI NETS 2014, Tunis, Tunisia, June 23-27. (2014) 210–232
8. Weidlich, M., Mendling, J., Weske, M.: Efficient consistency measurement based on behavioral profiles of process models. IEEE Tr. Soft. Eng. **37**(3) (2011) 410–429
9. Weidlich, M., Polyvyanyy, A., Mendling, J., Weske, M.: Causal behavioural profiles - efficient computation, applications, and evaluation. Fundam. Inform. **113**(3-4) (2011) 399–435
10. Schoknecht, A., Thaler, T., Fettke, P., Oberweis, A., Laue, R.: Similarity of business process models: A state-of-the-art analysis. ACM Comput. Surv. **50**(4) (August 2017) 52:1–52:33
11. Miller, G.A.: Wordnet: a lexical database for english. Communications of the ACM **38**(11) (1998) 39–41
12. Leopold, H., Smirnov, S., Mendling, J.: Refactoring of process model activity labels. In: Natural Language Processing and Information Systems. Springer (2010) 268–276
13. Van der Aa, H., Leopold, H., del Rio-Ortega, A., Resinas M., Reijers, H.A.: Transforming unstructured natural language descriptions into measurable process performance indicators using hidden markov models. Information Systems (2017)
14. Padró, L., Stanilovsky, E.: Freeling 3.0: Towards wider multilinguality. In: Proceedings of the Language Resources and Evaluation Conference (LREC 2012), Istanbul, Turkey, ELRA (May 2012)
15. Torras, C.: Relaxation and neural learning: Points of convergence and divergence. Journal of Parallel and Distributed Computing **6** (1989) 217–244
16. Daudé, J., Padró, L., Rigau, G.: Mapping wordnets using structural information. In: 38th Annual Meeting of the Association for Computational Linguistics (ACL'2000)., Hong Kong (2000)
17. Sànchez-Ferreres, J., Carmona, J., Padró, L.: Aligning textual and graphical descriptions of processes through ILP techniques. In: Advanced Information Systems Engineering - 29th International Conference,CAiSE 2017, Essen, Germany, June 12-16, 2017, Proceedings. (2017) 413–427
18. Becker, M., Laue, R.: A comparative survey of business process similarity measures. **63** (02 2012) 148–167
19. : Process model matching contest 2015. https://ai.wu.ac.at/emisa2015/contest.php Accessed: 2017-11-13.