Structural Computation of Alignments of Business Processes over Partial Orders

Farbod Taymouri[1] Josep Carmona[2]

[1]The University of Melbourne

[2]Universitat Politècnica de Catalunya

27/06/2019
Aachen, Germany
The 19th International Conference on Application of Concurrency to System Design

Agenda

Introduction

Related Work

Challenges and Objectives

Preliminaries

An Example

Overall Framework
- It. Optimization
- Dynamic Programming

Experiments

Conclusion and Future Work

Introduction

- *Conformance Checking* : Is a set of techniques that aim to identify deviations between a process model and its footprint. It fundamentally boils down to the **Alignment notion**.

FIGURE – Alignment notion

Introduction

- Example :

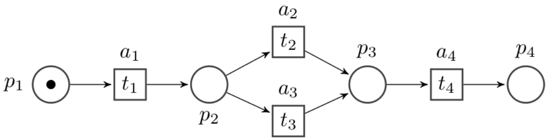


FIGURE – Process model in WF-net

Observed trace : $\sigma = a_1 a_1 a_4 a_2$

$\alpha =$

a_1	a_1	$\perp$	a_4	a_2
t_1	$\perp$	t_3	t_4	$\perp$

,

- Synchronous move
- Asynchronous move

Introduction

- Example :

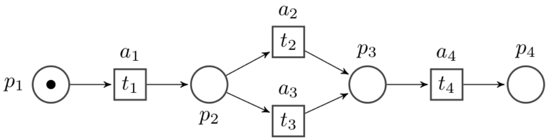


FIGURE – Process model in WF-net

Observed trace : $\sigma = a_1 a_1 a_4 a_2$

$\alpha=$

a_1	a_1	$\perp$	a_4	a_2
t_1	$\perp$	t_3	t_4	$\perp$

, $\alpha=$

a_1	a_1	$\perp$	a_4	a_2
$\perp$	t_1	t_2	t_4	$\perp$

- Synchronous move
- Asynchronous move

Related Work

- *Optimal Alignment Computation*
  1. Approaches based on A^* [A. Adriansyah, 2014]. **State of the art approach**.
  2. Automata based approach [D. Reissner et al., 2017], [S. J. J. Leemans et al., 2018]. **State of the art approach**.
  3. Automated planning approach [M. de Leoni et al., 2017].
- *Approximate to Optimal Alignment Computation*
  1. Structural theory of Petri net and ILP based approach [F. Taymouri et al., 2016].
  2. Hybrid based approach (Structural theory of Petri net, ILP and A^*) [B. Van Dongen et al., 2017].
  3. Incremental approach (Structural theory of Petri net, ILP and A^*) [B. Van Dongen et al., 2018].
  4. Decision diagrams [V. Bloemen et al., 2018, (ACSD), (BPM)].
  5. Evolutionary approach (Structural theory of Petri net, ILP and G.A.) [F. Taymouri et al., 2018].

Challenges and Objectives

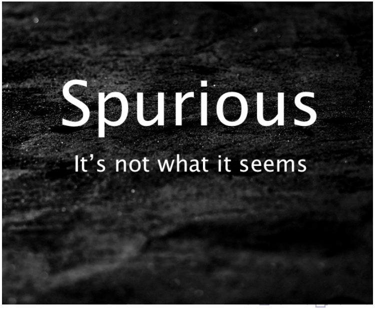
The main challenges ahead are as follows :

- Approaches based structural theory of Petri nets, *though scalable*, suffers from **Spurious** solutions

The main goal of this research is as follows :

- Computing an alignment based on structural theory of Petri nets that is **free from spurious solutions.**

$$\textit{Marking Equation} + \textit{Acyclic net} \rightarrow \textit{No Spuriuos Solution}$$



Preliminaries

- Process models are represented by **labeled WF-nets**, (N, m_{start}, m_{end}).

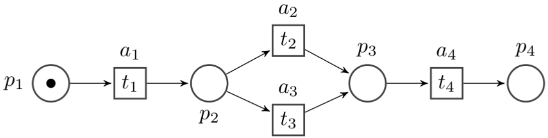


FIGURE – WF-net

$\ell(t_1) = a_1,\ \ell(t_2) = a_2,\ \ell(t_3) = a_3,\ \ell(t_4) = a_4$

- **Marking equation** shows required transitions need to be fired from one marking to another one.

$$m_i = m_{start} + \mathbf{N.X} \tag{1}$$

$$\begin{matrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{matrix} \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} + \overset{\begin{matrix} t_1 & t_2 & t_3 & t_4 \end{matrix}}{\begin{pmatrix} -1 & 0 & 0 & 0 \\ 1 & -1 & -1 & 0 \\ 0 & 1 & 1 & -1 \\ 0 & 0 & 0 & 1 \end{pmatrix}} . \begin{matrix} t_1 \\ t_2 \\ t_3 \\ t_4 \end{matrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Preliminaries

- A finite and complete **unfolding prefix** π of a Petri net N is a finite acyclic net which implicitly represents all the **reachable states** of N, together with **transitions enabled at those states**.
- Respective elements of a π are called **event**, B, and **condition**, E, **cut-off event**, $E_{cut} \subseteq E$, and $\rho : B \cup E \rightarrow P \cup T$.

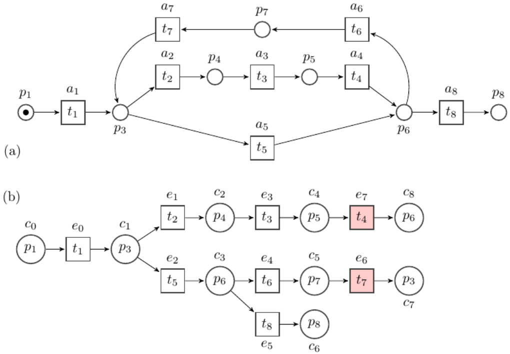


FIGURE – (a) WF-net, (b) Unfolding Prefix π

Preliminaries

- An **observed trace** is a sequence of events, i.e., $\sigma_1 = a_1a_2a_4$, $\sigma_2 = a_1a_1a_3a_4$.
- A **modeled trace** is a sequence of transitions of the given model, i.e., $\sigma_{N_1} = t_1t_3t_4$, $\sigma_{N_2} = t_1t_3t_2t_4$.
- Given an alphabet, a **Parikh vector** of a trace shows the occurrence of each element, i.e.,

$$\widehat{\sigma_{N_1}} = \begin{matrix} t_1 \\ t_2 \\ t_3 \\ t_4 \end{matrix} \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \end{pmatrix}, \widehat{\sigma_2} = \begin{matrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{matrix} \begin{pmatrix} 2 \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

Preliminaries

- **Theorem** (Marking Equation for Acyclic Petri nets) : Let N be an acyclic Petri net. If the vector y satisfies the marking equation :

$$m_i = m_0 + \mathbf{N}.\mathbf{X}$$

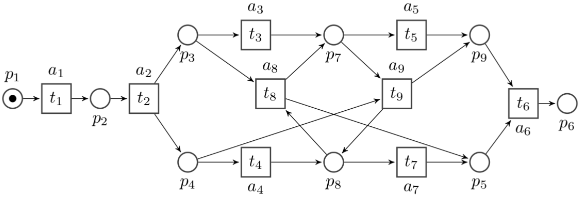
Then there exists a **firing sequence** σ firable from marking m_0 such that $y = \widehat{\sigma}$, [A. Giua et al., 2007].

Motivational Example

The following example shows how the use of the marking equation and a WF-net that result in a spurious solution.

- Example : Given an observed trace $\sigma = a_1 a_2 a_8 a_9 a_6$, *which contains deviations*, it is possible to have an alignment with fitness value 1 and not executable!!

$$\alpha = \begin{array}{|c|c|c|c|c|} a_1 & a_2 & a_8 & a_9 & a_6 \\ \hline t_1 & t_2 & t_8 & t_9 & t_6 \end{array}$$



Overall Framework

Proposing an iterative optimization formulation for the alignment computation.

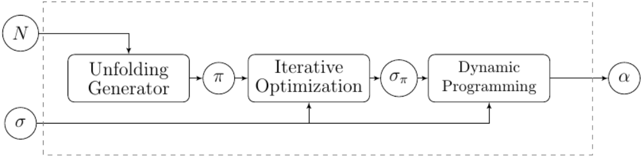


FIGURE – The overall framework of the proposed technique

It must be noted that the unfolding prefix is computed only once, for the whole event log.

Iterative Optimization

In general, *as an optimization problem*, we are looking for a Parikh vector X, that has **maximum similarity** to the observed Parikh trace, i.e., $\widehat{\sigma}$ [Taymouri et al., 2016]. However, it needs to be modified due to the following reasons :

- The unfolding prefix π does not have one sink compared to its WF-net, and it might represents multiple sinks accordingly.
- Transitions and places might be represented multiple times in the unfolding prefix.

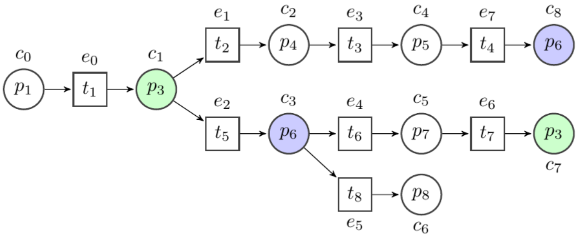


FIGURE – The same elements are highlighted

Iterative Optimization (cont.)

In more details the following optimization instance alongside with additional constraints must be solved :

- Modified version of [Taymouri et al., 2016], i.e., relaxing the constraint that marks the final marking of WF-net (since there is no one unique sink in π) :

$$
\begin{aligned}
&\text{Maximize} \\
&\Big(\sum_{\ell(e)\in J} X[e] - \delta \times \sum_{\ell(e)\notin J} X[e] + 0 \times \sum_{\ell(e)=\tau} X[e]\Big), \\
&\text{Subject to:} \\
&m_i = m_j + \mathbf{N}_\pi . X \\
&\forall e \in X, \forall a \in \hat{\sigma} \quad \text{If} \quad \ell(e) \in J \quad \text{and} \quad \ell(e) = a : \\
&\widehat{\sigma}[a] = \sum_{\ell(e)=a} (X[e] + X^s[e]), \\
&\sum_{\forall \ell(e)\in E_{cut}} X[e] \geq 1, \\
&X, X^s \geq \mathbf{0}
\end{aligned}
$$

Iterative Optimization (cont.)

In more details the following optimization instance alongside with additional constraints must be solved :

- To preserve the semantic of of firing rules and more importantly to allow the iteration, *recharging duplicate places*, the following constraint is added :

$$\sum_{b_k \in B(b_k)=p_i} b_k = 1$$

- To force the firing of cut-ff events, E_{cut}, marking the final sink in the original WF-net, the following terms will be added to the objective function in the next iterations :

$$\sum_{\forall \ell(e) \in E_{cut}} X[e] + \sum_{e \in path} X[e]$$

path is the set of transitions on the branch in π that is from $B(b_i) = m_{start}$ to $B(b_i) = m_{end}$.

Iterative Optimization (cont.)

How does it work in practice ?

- The previous optimization problem must be solved multiple times. In each iteration the input is updated, i.e., only remaining events in $\widehat{\sigma}$ are fed into the algorithm or $\forall_{e \in J} \widehat{\sigma_{k+1}}[e] = \widehat{\sigma_k}[e] - X_k[e]$.
- It iterates until convergence, i.e., no more events are remaining in the Parikh observed trace.
- The result is an ordered set of Parikh vectors, i.e., $X_1 X_2 \ldots X_k$. For each X_i, the respective elements has an order [1]. Thus, the ordered set of Parikh vectors constitutes a modeled trace.

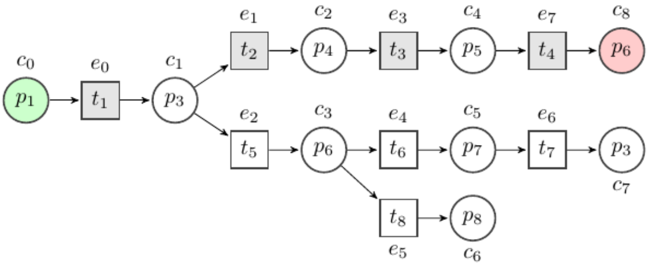
1. Lemma 4.2 in the paper

Iterative Optimization (cont.)

Example : Consider the following model and the observed trace $\sigma = a_1 a_3 a_2 a_6 a_7 a_8$

First

$$\hat{\sigma_1} = \begin{matrix} t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \\ t_7 \\ t_8 \end{matrix} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \end{pmatrix} \quad X_1 = \begin{matrix} t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \\ t_7 \\ t_8 \end{matrix} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

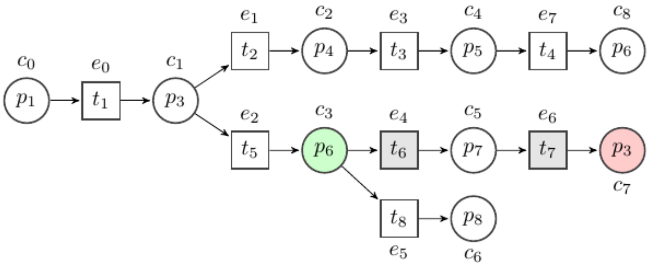


Iterative Optimization (cont.)

Example : Consider the following model and the observed trace $\sigma = a_1 a_3 a_2 a_6 a_7 a_8$

Second

$$\hat{\sigma_2} = \begin{matrix} t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \\ t_7 \\ t_8 \end{matrix} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{pmatrix} \quad X_2 = \begin{matrix} t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \\ t_7 \\ t_8 \end{matrix} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

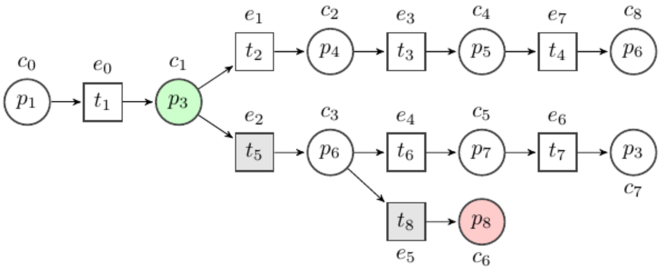


Iterative Optimization (cont.)

Example : Consider the following model and the observed trace $\sigma = a_1 a_3 a_2 a_6 a_7 a_8$

Third

$$\hat{\sigma_3} = \begin{array}{c} t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \\ t_7 \\ t_8 \end{array} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \quad X_3 = \begin{array}{c} t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \\ t_7 \\ t_8 \end{array} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$



Iterative Optimization (cont.)

Example : Consider the following model and the observed trace $\sigma = a_1 a_3 a_2 a_6 a_7 a_8$

$$X_1 = \begin{matrix} t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \\ t_7 \\ t_8 \end{matrix} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad X_2 = \begin{matrix} t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \\ t_7 \\ t_8 \end{matrix} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} \quad X_3 = \begin{matrix} t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \\ t_7 \\ t_8 \end{matrix} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

$$\Rightarrow \quad \sigma_N = t_1 t_2 t_3 t_4 t_6 t_7 t_5 t_8$$

At this time, we have an executable model sequence σ_N that can be aligned with the observed trace σ ! This can be done using the well-known dynamic programming technique for aligning two sequences.

Alignment Computation Using Dynamic Programming (A small example)

This stage aligns two sequences, i.e., modeled and observed tarce, that is inspired from [S. B. Needleman et al., 1970] and presented in [F. Taymouri et al., 2016].

- Assume $\sigma = a_1 a_4 a_6$ and $\sigma_N = t_1 t_4 t_5 t_6$ with $\ell(t_1) = a_1, \ell(t_2) = a_2,$ $\ell(t_3) = a_3, \ell(t_4) = a_4.$
- Create and initialize a matrix and filling it with the following formula :

		a_1	a_4	a_6
	0	-1	-2	-3
t_1	-1			
t_4	-2			
t_5	-3			
t_6	-4			

$$SIM(t_i, a_j) = MAX \begin{cases} SIM(t_{i-1}, a_{j-1}) + s(t_i, a_j) \\ SIM(t_{i-1}, a_j) - \delta \\ SIM(t_i, a_{j-1}) - \delta \end{cases} \qquad s(t_i, a_j) = \begin{cases} \beta & \text{If} \quad \ell(t_i) = a_j \\ -\beta & \text{If} \quad \ell(t_i) \neq a_j \end{cases}$$

Alignment Computation Using Dynamic Programming (A small example)

This stage aligns two sequences, i.e., modeled and observed tarce, that is inspired from [S. B. Needleman et al., 1970] and presented in [F. Taymouri et al., 2016].

- Assume $\sigma = a_1 a_4 a_6$ and $\sigma_N = t_1 t_4 t_5 t_6$ with $\ell(t_1) = a_1, \ell(t_2) = a_2,$ $\ell(t_3) = a_3, \ell(t_4) = a_4.$
- After filling the matrix, traceback to obtain the alignment

		a_1	a_4	a_6
	0	-1	-2	-3
t_1	-1	1	0	-1
t_4	-2	0	2	1
t_5	-3	-1	1	0
t_6	-4	-2	0	2

$\alpha =$

a_1	a_4	$\perp$	a_6
t_1	t_4	t_5	t_6

Experiments

A prototype was developed in Python 2.7 and Gurobi [2] was used as the ILP solver. Unforlding prefixes were generated using Punf [3] [V. Khomenko et al., 2001]. Various datasets, realistic and artificial, were used for the evaluation.

2. www.gurobi.com
3. http://homepages.cs.ncl.ac.uk/victor.khomenko/tools/punf/

Experiments

- *Ratio Size* : Ratio between WF-nets and corresponding Unfolding prefixes

Model	$\frac{\lvert P_{un.}\rvert}{\lvert P_{or.}\rvert}$	$\frac{\lvert T_{un.}\rvert}{\lvert T_{or.}\rvert}$	$\frac{\lvert Arc_{un.}\rvert}{\lvert Arc_{or.}\rvert}$
prAm6	422/363	363/343	842/846
prBm6	317/317	375/317	748/752
prCm6	317/317	375/317	748/752
prDm6	569/529	429/429	1136/1140
prEm6	325/277	275/275	648/652
prFm6	385/362	299/299	768/772
prGm6	412/357	335/335	822/826
M_1	47/ 40	39/39	92/92
M_2	41/34	34/34	80/80
M_3	139/108	123/123	276/276
M_4	54/36	52/52	106/106
M_5	40/35	33/33	78/78
M_6	85/69	72/72	168/168
M_7	75/65	62/62	148/148
M_8	19/17	15/15	36/36
M_9	61/47	55/55	120/120
M_{10}	178/150	146/146	354/354
ML_1	38/27	35 /35	74/74
ML_2	203/165	177/177	404/404
ML_3	54/45	45/45	106/106
ML_4	41/36	33/33	80/80
ML_5	196/159	172/172	390/390
Road.	25/15	23/23	48/48
Bank.	137/121	114/114	272/272

Experiments

- *Average Iteration* : Average number of iterations for a modeled trace computation until converging.

Model	$\overline{It.}$	Model	$\overline{It.}$
prAm6	4.15	M1	3.96
prBm6	5.07	M2	4.14
prCm6	5.08	M3	10.27
prDm6	28.42	M4	7.49
prEm6	10.34	M5	5.92
prFm6	15.61	M6	10.42
prGm6	9.49	M7	8.39
Bank.	8.11	M8	3.81
ML1	5.14	M9	4.44
ML2	20.11	M10	10.27
ML3	9.54	ML4	9.67
ML5	2.78	Road.	8.18

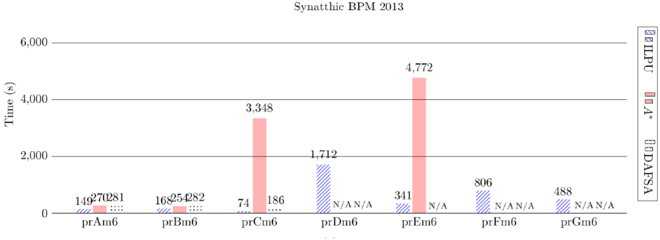
Experiments

- *Percentage of reproduced events* : The percentage of observed events that the approach reproduces in average for an input σ.

Model	Unfolding	$A*$	Model	Unfolding	$A*$
prAm6	0.91	0.95	M1	1.00	0.77
prBm6	0.84	1.00	M2	1.00	0.71
prCm6	0.86	0.71	M3	0.98	0.92
prDm6	0.89	NA	M4	0.92	0.50
prEm6	0.78	0.98	M5	0.98	0.77
prFm6	0.95	NA	M6	0.98	NA
prGm6	0.83	NA	M7	0.97	NA
Bank.	0.78	0.99	M8	1.00	0.71
ML1	0.69	0.63	M9	0.63	0.73
ML2	0.96	NA	M10	0.95	NA
ML3	0.97	0.47	ML4	0.99	0.48
ML5	0.90	NA	Traffic.	0.68	0.58

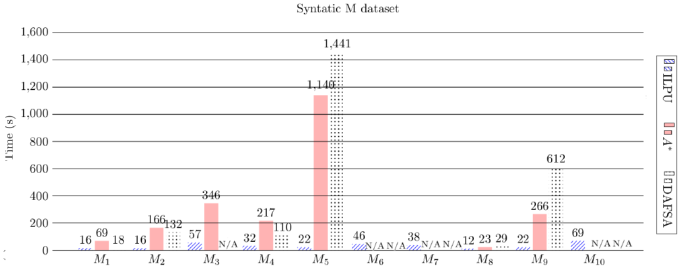
Experiments

- *Execution time* :



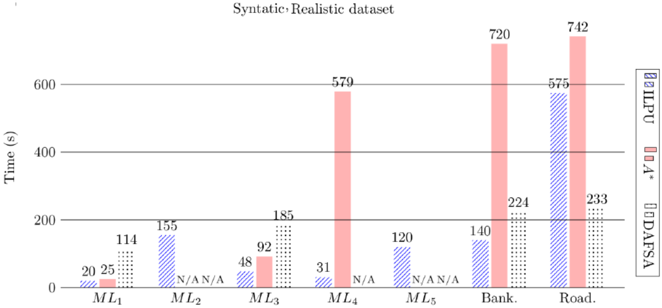
Experiments

- *Execution time* :



Experiments

- *Execution time* :



Conclusion and Future Work

- A novel optimization approach for the alignment computation based on the structural theory of Petri nets, ILP, and Unfolding of Petri nets is proposed that is free from **spurious solutions**.
- Though the experiments witness the merit of the proposed approach, i.e., the obtained sequence is executable, the corresponding quality, in terms of events' order, needs to be improved for models with **high degree of concurrency**. Above that, the input model must be **safe** for the unfolding computation.
- The proposed approach can be integrated as a heuristic for the other alignment computation approaches to speed up their computations.


Thank you