ON THE DISRUPTIVE EFFECTIVENESS OF AUTOMATED PLANNING FOR LTL_f-BASED TRACE ALIGNMENT

REASONING AGENTS PROJECT

Alessia Carotenuto 1764400
Emanuele Iacobelli 1657799
Veronica Romano 1580844

SUMMARY

- Introduction
- "On the Disruptive Effectiveness of Automated Planning for LTLf -based Trace Alignment" [De Giacomo, Maggi, Marrella, Patrizi. 2017]
    - How to define Business Processes and DECLARE models
    - The logic LTL_f and the associated NFA
    - The Trace Alignment Problem
    - Trace Alignment as Planning and how to implement it in PDDL
    - Experiment: Real and synthetic logs
- Our implementations
- Comparison between the results (original work VS our implementations)

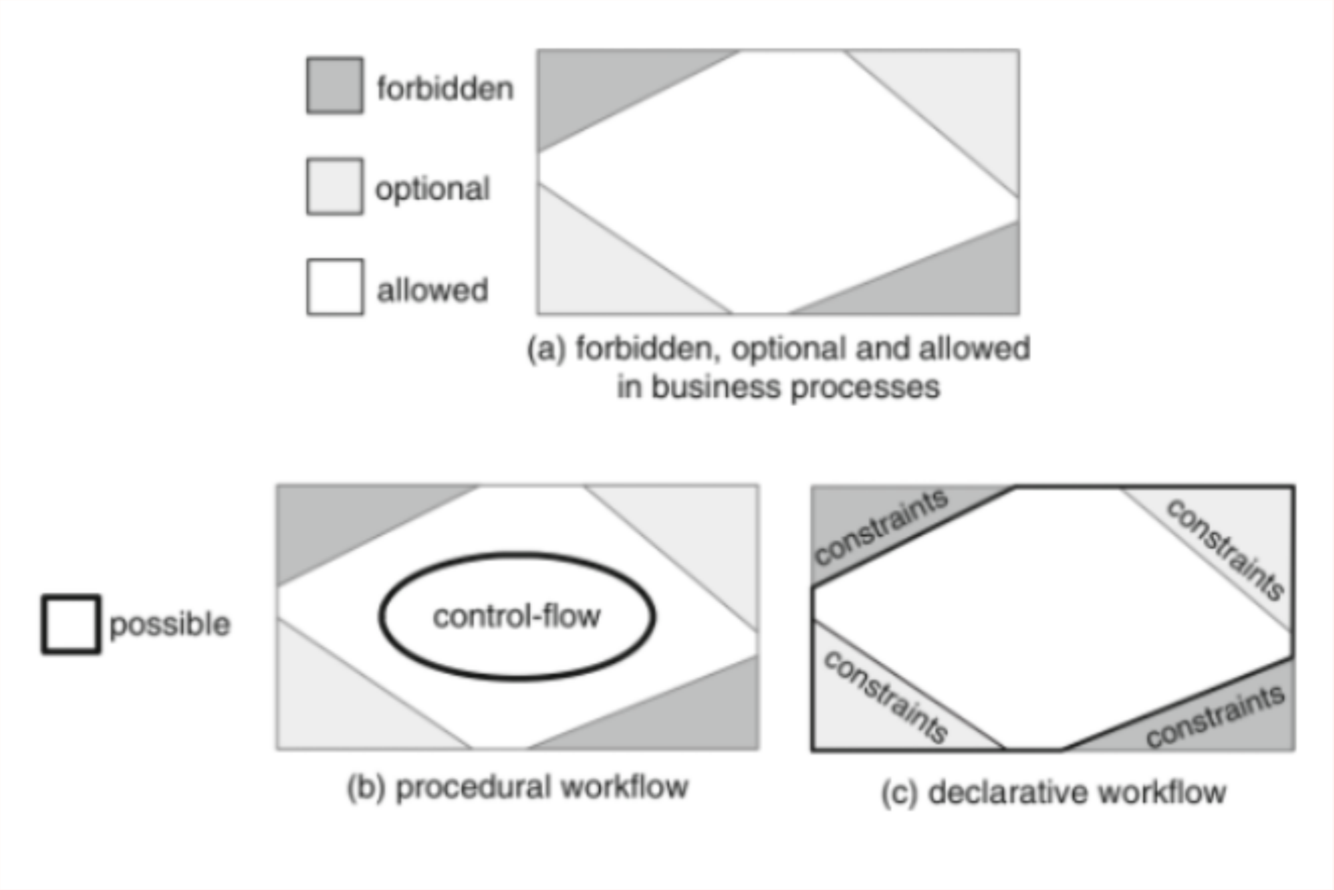
INTRODUCTION

- Trace Alignment problem and BPM
  - "The Trace Alignment is the problem of "cleaning" and "repairing" dirty traces to make them compliant with the underlying process model" [van der Aalst, 2016]
- In the paper [De Giacomo, Maggi, Marrella, Patrizi. 2017] is proposed a technique to synthesize the alignment instructions relying on finite automata theoretic manipulations: an implementation of this technique can be effectively implemented by using the cost optimal planning
- Original work: the implementation described in [De Giacomo, Maggi, Marrella, Patrizi. 2017] is compared with ad hoc alignment systems [de Leoni, Maggi, and van der Aalst 2012; 2015] and previous approaches based on classical planning [De Giacomo et al. 2016]

BUSINESS PROCESS

- BPM works a lot with high-level processes that are used to organize finite tasks at a high-level
  - Example:

    GO TO THE OFFICE → PICK A BOOK → GO TO THE BOSS' OFFICE → LEAVE THE BOOK →GO TO THE OFFICE ...



- PROCEDURAL APPROACH VS DECLARATIVE APPROACH

THE LOGIC LTL$_f$ AND …

- **LINEAR TEMPORAL LOGIC (LTL) → LINEAR TEMPORAL LOGIC OVER FINITE TRACES (LTL$_f$)**
  - **Why using LTL$_f$ instead of LTL?**
  - **Different syntax:**

$$\varphi ::= A \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \bigcirc\varphi \mid \varphi_1\,\mathcal{U}\,\varphi_2$$

- A: atomic propositions
- $\neg\varphi$, $\varphi_1 \wedge \varphi_2$: boolean connectives
- $\bigcirc\varphi$: "next step exists and at next step (of the trace) φ holds"
- $\varphi_1\,\mathcal{U}\,\varphi_2$:"eventually φ_2 holds, and φ_1 holds until φ_2 does"
- $\bullet\varphi \doteq \neg\bigcirc\neg\varphi$: "if next step exists then at next step φ holds" *(weak next)*
- $\Diamond\varphi \doteq \mathtt{true}\,\mathcal{U}\,\varphi$: "$\varphi$ will eventually hold"
- $\Box\varphi \doteq \neg\Diamond\neg\varphi$: "from current till last instant φ will always hold"
- $Last \doteq \neg\bigcirc\mathtt{true}$: denotes last instant of trace.

For example, if we consider the following formula "$\neg X(\phi)$":

- **In LTL it means "In the next step $\neg\phi$ holds"**
- **In LTLf it means "Not exist the next instant or $\neg\phi$ holds there".**

... THE ASSOCIATED NFA

Each LTL$_f$ formula has an associated NFA which accepts exactly all the traces that satisfy the specific formula from which is created

- **NFA: automata in which from a state may exist more than one transition to another state with the same input symbol**

$$A = < \Sigma, Q, q_0, \rho, F >$$

Where:

- Σ **is the input alphabet**
- Q **is the finite set of automaton states**
- $q_0 \in Q$ **is the initial state**
- $\rho \subseteq Q \times L_{Prop} \times Q$ **is the transition relation**
- $F \subseteq Q$ **is the set of the final states**

THE TRACE ALIGNMENT PROBLEM

- **A log trace, written as $t = a \ b \ c$, is a trace such that the propositional interpretation associated with each position contains only one proposition**
- **FORMULATION OF THE PROBLEM: given a trace t and an LTL$_f$ formula φ such that $t \not\models \varphi$, find a trace $\hat{t}$ such that $\hat{t} \models \varphi$ and "$cost(t, \hat{t}) = c$" is minimal**
- **SOLUTION: the trace t has to be modified by using the lowest number of repairing actions ("addition" and deletion")**

THE TRACE ALIGNMENT PROBLEM

- If φ is satisfiable, a solution for the problem always exists because starting from the initial trace, by using the repairing actions, it is possible to generate any log trace
- Even in this case we can create an automaton which represents the trace T; by using this latter and A we can address the Trace Alignment Problem
- Given the trace $t = e_1 \dots e_n$, the constraint φ and it's corresponding NFA (constraint automaton) $A = < \Sigma, Q, q_0, \rho, F >$, the DFA (trace automaton) is $T = < \Sigma_t, Q_t, q_0^t, \rho_t, F_t >$ where:
  - $\Sigma_t = \{e_1, \dots, e_n\}$
  - $Q_t = \{q_0^t, \dots, q_n^t\}$ is a set of $n+1$ arbitrary states
  - $\rho_t = \cup_{i=0,\dots,n-1} < q_i^t, e_{i+1}, q_{i+1}^t >$
  - $F_t = \{q_n^t\}$

THE TRACE ALIGNMENT PROBLEM

- But before solving this problem, T and A has to be augmented in order to accept traces modified with the repairing actions:
  - $T^+ = < \Sigma_t^+, Q_t, q_0^t, \rho_t^+, F_t >$
  - $A^+ = < \Sigma^+, Q, q_0, \rho^+, F >$
- Where:
  - $\Sigma_t^+ = \Sigma^+$: contains all the propositions in Σ_t, a proposition del_p for all propositions $p \in \Sigma$ and a proposition add_p for all proposition $p \in \Sigma \cup \Sigma_t$
  - ρ_t^+ contains all the transitions in ρ_t, a new transition $< q, del\_p, q' >$ for all transitions $< q, p, q' > \in \rho_t$ and a new transition $< q, add\_p, q >$ for all propositions p in Σ_t and states $q \in Q_t$ if there is no transition $< q, p, q' > \in \rho_t$ (for all $q' \in Q_t$)
  - ρ^+ contains all the transitions in ρ, a transition $< q, del\_p, q >$ for all $q \in Q$ and $p \in \Sigma_t$ and a transition $< q, add\_p, q' >$ for all transitions $< q, \psi, q' > \in \rho$ such that $p \models \psi$
  - Q_t, q_0^t and F_t are the same of the automaton T
  - Q, q_0 and F are the same of the automaton A

THE TRACE ALIGNMENT PROBLEM

- **Theorem:**
  - *Consider a log trace t and an LTL$_f$ formula φ, both over $Prop$, s.t. $t \not\models \varphi$. Let T^+ and A^+ be the automata obtained from t and φ, as described above. If t^+ is a trace accepted by both A^+ and T^+ containing a minimal number of repair propositions (with respect to all other traces accepted by A^+ and T^+), then a trace $\hat{t}$ with minimal cost $cost(t, \hat{t})$ s.t. $\hat{t} \models \varphi$ can be obtained from t^+ by removing all propositions of the form del_p and replacing all propositions of the form add_p with p*

TRACE ALIGNMENT AS PLANNING AND HOW TO IMPLEMENT IT IN PDDL

- **DETERMINISTIC PLANNING DOMAIN:** $D = < S, A, C, \tau >$ **where:**
  - $S \subseteq 2^{Prop}$ **is the finite set of domain states**
  - A **is the finite set of domain actions**
  - $C : A \rightarrow \mathbb{N}^+$ **is a cost function**
  - $\tau : S \times A \rightarrow S$ **is the transition function**
- **COST-OPTIMAL PLANNING PROBLEM:** $P = < D, s_0, G >$ **where:**
  - D **is a planning domain with action costs**
  - $s_0 \in S$ **is the initial state of the problem**
  - G **is the problem goal**

TRACE ALIGNMENT AS PLANNING AND HOW TO IMPLEMENT IT IN PDDL

- **Trace Alignment problem as a cost-optimal problem $D = < S, A, C, \tau >$ is defined as:**
  - $S \subseteq 2^{Q_t \cup Q}$**: automata states**
  - $A = \{sync\_e, del\_e, add\_e \,|\, e \in \Sigma \cup \Sigma_t\}$ **is the set of the repair propositions used as actions**
  - **For all** $e \in \Sigma \cup \Sigma_t$, $C(sync\_e) = 0$ **and** $C(del\_e) = C(add\_e) = 1$
  - τ **is defined as follow for all** $e \in \Sigma \cup \Sigma_t$, $q_t, q_t' \in Q_t$ **and** $R, R' \subseteq Q$**:**
    - $\tau(\{q_t\} \cup R, sync\_e) = \{q_t'\}$ **iff** $q_t \xrightarrow{e} q_t' \in \rho_t^+$ **and for all** $q \in R$ **and** $q' \in R'$ **there exists** ψ **s.t.** $e \models \psi$ **and** $q \xrightarrow{\psi} q' \in \rho^+$
    - $\tau(\{q_t\} \cup R, del\_e) = \{q_t'\} \cup R'$ **iff** $q_t \xrightarrow{del\_e} q_t' \in \rho_t^+$ **and for all** $q \in R$ **and** $q' \in R', q \xrightarrow{del\_e} q' \in \rho^+$
    - $\tau(\{q_t\} \cup R, add\_e) = \{q_t'\} \cup R'$ **iff** $q_t \xrightarrow{add\_e} q_t' \in \rho_t^+$ **and for all** $q \in R$ **and** $q' \in R', q \xrightarrow{add\_e} q' \in \rho^+$
- **This formulation for the domain allows to move synchronously on T^+ and A^+. So when the final states of T^+ and A^+ are reached, the repairing trace t^+ satisfies φ. Since a plan with minimal cost is required, the planning problem $P = < D, s_o, G >$ is defined as:**
  - $s_0 = \{q_0, q_0^t\}$
  - $G = q_t^f \wedge \bigvee_{q \in F} q$, **for** $q_t^f \in F_t$

PLANNING DOMAIN

- **Two abstract types:**
  - *activity*
  - *state*
    - *automaton_state*
    - *trace_state*
- **Three actions:**
  - *sync*
  - *add*
  - *del*
- **Four predicates:**
  - *(trace ?t1 - trace_state ?e - activity ?t2 - trace_state)*
  - *(automaton ?s1 - trace_state ?e - activity ?s2 - trace_state)*
  - *(cur_state ?s - state)*
  - *(final_state ?s - state)*
- **A numeric fluent:**
  - *total-cost*

ACTIONS

```
(:action sync
      :parameters (?t1 - trace_state ?e - activity ?t2 - trace_state)
      :precondition (and  (cur_state ?t1)     (trace ?t1 ?e ?t2))
      :effect (and   (not (cur_state ?t1))
                     (cur_state ?t2)
                     (forall   (?s1 ?s2 - automaton_state)
                          (when     (and (cur_state ?s1) (automaton ?s1 ?e ?s2) )
                                    (and (not (cur_state ?s1) ) (cur_state ?s2) )
                     )
                )
           )
)

(:action add
      :parameters (?e - activity)
      :precondition (and )
      :effect (and   (increase (total-cost) 1 )
                     (forall (?s1 ?s2 - automaton_state)
                          (when     (and (cur_state ?s1) (automaton ?s1 ?e ?s2) )
                                    (and (not (cur_state ?s1)) (cur_state ?s2))
                          )
                     )
           )
)

(:action del
      :parameters (?t1 - trace_state ?e - activity ?t2 - trace_state)
      :precondition (and (cur_state ?t1) (trace ?t1 ?e ?t2) )
      :effect (and (increase (total-cost) 1 ) (not (cur_state ?t1) ) (cur_state ?t2) )
)
```

PLANNING PROBLEM

It contains:

- **All the objects needed to describe the trace automaton and the constraint automata (states and activities)**
- **The initialization of the objects at their initial conditions (the transitions of all the automata, the initial states and the final states)**
- **The goal to be reached expressed:** $G = q_t^f \wedge \bigvee_{q \in F} q$ **for** $q_t^f \in F_t$

```
(:goal (and (cur_state t33) (forall (?s - automaton_state) (imply (cur_state ?s)
(final_state ?s)) )) )
```

OUR IMPLEMENTATIONS

- **PDDL2.1**

Two different versions of the encoding:

1. **Encoding provided in the original paper**
2. **Modified version of encoding 1:**
   - **Presence of dummy states as final states**
   - **Conjunction in the goal of the problem files**

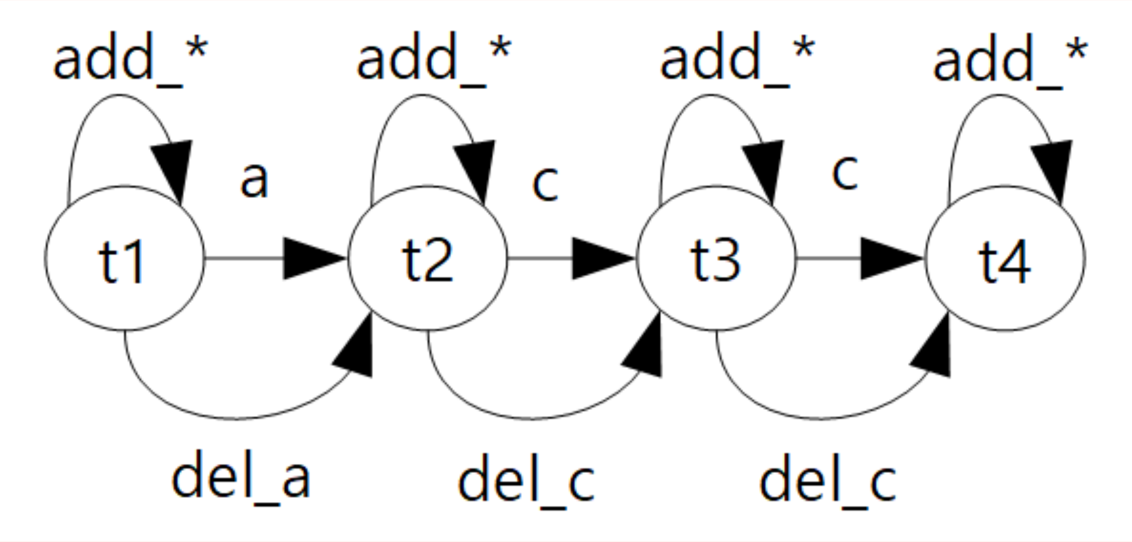
ENCODING PROVIDED IN THE ORIGINAL PAPER

- **DOMAIN FILE: unique for all the problems and written by hand**
- **PROBLEM FILES: generated automatically from the script** *"createProbPddlDummyFree.py"*
  - **Objects: all the states of the trace automaton correlated to the problem, all the states of the automata associated with the constraints and all the activities involved in both $T^+$and all the A^+**
  - **Initial conditions: the initial state of T^+ and all the initial states of the $A_1^+, \ldots, A_n^+$, the final state of T^+ and all the final states of $A_1^+, \ldots, A_n^+$ and all the transitions of both the trace and the constraint automata**
  - **Goal conditions: the "AND" between the final state of T^+ as current state and a *forall* that checks if the current states of the constraint automata imply final states**

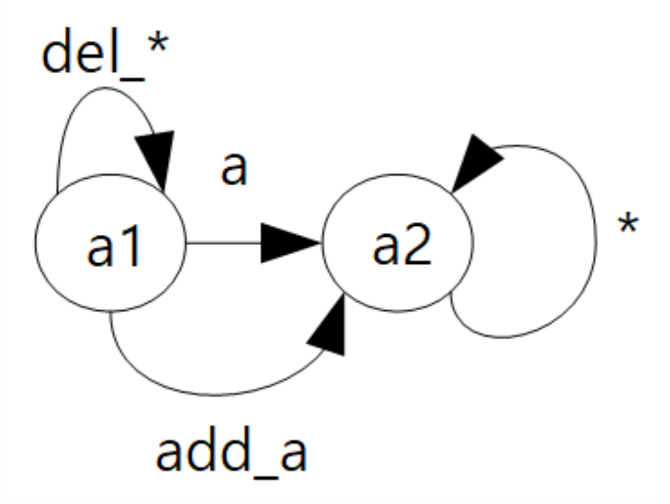
```
(:goal (and (cur_state t33) (forall (?s - automaton_state) (imply (cur_state ?s)
(final_state ?s)) )) )
```

AN EXAMPLE

- $t = a\ c\ c$



- $\varphi = F\ a$



MANIPULATION OF THE TRANSITIONS OF THE DFA: LTLF2DFA AND MONA

```
---dfa---- DFA for formula with free variables: A B
Initial state: 0
Accepting states: 1
Rejecting states: 0 2

Automaton has 3 states and 4 BDD-nodes
Transitions:
State 0: XX -> state 1
State 1: 0X -> state 1
State 1: 10 -> state 2
State 1: 11 -> state 1
State 2: X0 -> state 2
State 2: X1 -> state 1
A counter-example of least length (1) is:
A               X 1
B               X 0

A = {0}
B = {}

A satisfying example of least length (0) is:
A               X
B               X

A = {}
B = {}
```

- ***State 0*** **and all the transitions starting from it are eliminated**
- **All the loops are eliminated**
- **For all the other transitions:**
  - **More than one "1": the transitions are eliminated**
  - **Only one explicit "1": the "X" in the transition are replaced with "0"**

MANIPULATION OF THE TRANSITIONS OF THE DFA

- Let's consider a longer transition like "0XXX", in which we consider to have the for this constraint automaton free variables A, B, C and D ad for the trace automaton the activities A, B, D and E

- There are no explicit "1" in the transition, so we have two possibilities:
  - Obtaining a transition with all "0" (no free variables involved in transition but the activity E from the trace can be executed)
  - Obtaining a number of transitions equal to the number of "X" (i.e. "0100", "0010", "0001")

MODIFIED VERSION OF THE ENCODING

- **DOMAIN FILE: unique for all the problems and written by hand**
- **PROBLEM FILES: generated automatically from the script** *"createProbPddl.py"*

TWO MODIFICATIONS:

- **FIRST MODIFICATION: implementation of the dummy states in the constraint automata, in order to reduce the number of final states which have to be checked every time**
- **SECOND MODIFICATION: change of the expression of the goal in such away to reduce the computational cost**

FIRST MODIFICATION

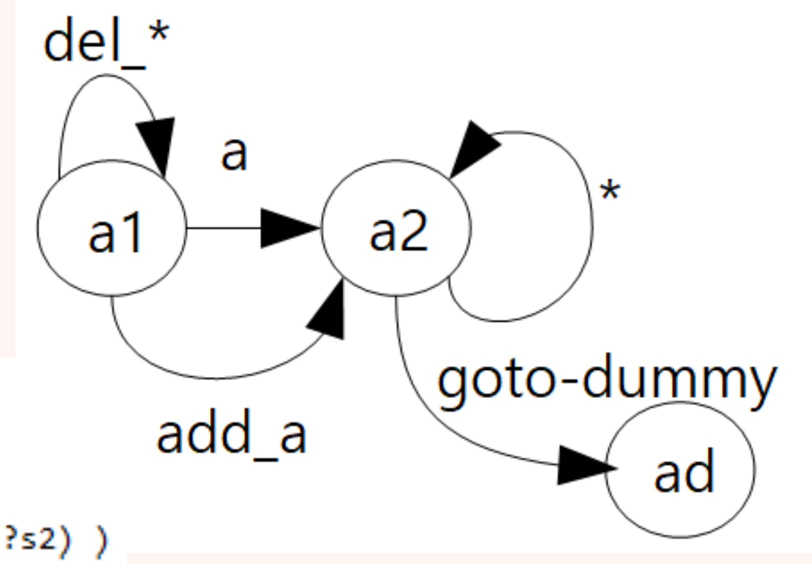
Has been added to the encoding 1:

- **A new abstract type "*dummy_activity*"**

```
(:types automaton_state trace_state - state  activity dummy_activity)
```

- ***A new proposition "(dummy ?s1 - automaton state ?e - dummy activity ?s2 -automaton state)"***

- ***A new action named "goto-dummy"***




```
(:action goto-dummy
      :parameters (?t - trace_state ?d - dummy_activity)
      :precondition (and (cur_state ?t) (final_state ?t) )
      :effect ( forall (?s1 ?s2 - automaton_state)
                        (when ( and (final_state ?s1) (cur_state ?s1) (dummy ?s1 ?d ?s2) )
                              (and     (not (cur_state ?s1) )
                                       (cur_state ?s2) (final_state ?s2)
                              )
                        )
              )
 )
```

SECOND MODIFICATION

- **In the original work the goal is considered as:** $G = q_t^f \wedge \bigvee_{q \in F} q$ **for** $q_t^f \in F_t$

```
(:goal (and (cur_state t33) (forall (?s - automaton_state) (imply (cur_state ?s)
(final_state ?s)) )) )
```

- **Now the goal is the conjunction of:**
  - **The final state of the trace automaton**
  - **All the dummy states**
  - **All the final states of the constraint automata which have only 1 final state**

```
(:goal (and (cur_state a1_d) (cur_state a2_1) (cur_state a3_d) (cur_state a4_d) (cur_state
a5_d) (cur_state a6_1) (cur_state a7_d) (cur_state a8_2) (cur_state a9_1) (cur_state a10_d)
(cur_state t33) ) )
```

EXTRACTION OF INFO FROM THE DATA

```
-<trace>
   <string key="concept:name" value="Synthetic trace no. 00"/>
  -<event>
     <string key="concept:name" value="activity 24"/>
     <string key="lifecycle:transition" value="complete"/>
     <date key="time:timestamp" value="2015-04-29T14:01:28.942+03:00"/>
   </event>
  +<event></event>
```

```
-<constraintdefinitions>
  -<constraint id="10" mandatory="true">
     <condition/>
     <name>precedence</name>
    -<template>
      +<description></description>
       <display>precedence</display>
       <name>precedence</name>
       <text>( ! ("B" ) U "A" ) \/ ([](!("B"))) /\ ! ("B" )</text>
```

```
-<constraintparameters>
  -<parameter templateparameter="1">
    -<branches>
       <branch name="activity 9-complete"/>
     </branches>
   </parameter>
  -<parameter templateparameter="2">
    -<branches>
       <branch name="activity 10-complete"/>
     </branches>
   </parameter>
 </constraintparameters>
```

DATASET OVERVIEW: OUR IMPLEMENTATION

Types of logs - formatted in XES standard and DECLARE models formatted in XML:

- **Real-life**
- **Synthetic**
  - **10 constraints VS 15 constraints**
    - **3 constraints modified**
    - **4 constraints modified**
    - **6 constraints modified**

EXPERIMENTS: OUR IMPLEMENTATION

- Implementation of the encoding proposed in the original work:
  - 10 constraints
    - 3, 4 and 6 modified const.
  - 15 constraints
    - 3 modified const.
- Implementation of the encoding proposed by us
  - 10 constraints
    - 3, 4 and 6 modified const.
  - 15 constraints
    - 3, 4 and 6 modified const.

PLANNING SYSTEM

- **Fast Downward**
- **SymBA*-2**
- **De Leoni et al.**

VS

FAST DOWNWARD

20.06

Blind $A*$

RESULTS COMPARED AND METRICS USED

- Original work VS Our implementation with the dummy state
  - 10 const.: 3, 4 and 6 const. modified
  - 15 const.: 3, 4 and 6 const. modified
- Our implementations: with dummy state VS without dummy state
  - 10 const.: 3 const. modified
  - 15 const.: 3 const. modified

- Total average time
- Average cost
- Trace length
  - 1-50
  - 51-100
  - 101-150
  - 151-200
  - 201-250

RESULTS: ORIGINAL WORK VS OUR IMPLEMENTATION WITH THE DUMMY STATE

Trace length	Align. cost	Average cost
1-50	1.77	1.79
51-100	2.11	2.13
101-150	3.03	3.04
151-200	3.79	3.81

10 constraints (3 const. modified): original work - our implementation

Trace length	Align. cost	Average cost
1-50	1.71	1.73
51-100	2.23	2.26
101-150	3.07	3.09
151-200	4.2	4.24
201-250		5.33

15 constraints (3 const. modified): original work - our implementation

Trace length	Align. cost	Average cost
1-50	2.74	2.77
51-100	5.86	5.88
101-150	9.68	9.7
151-200	13.42	13.42

10 constraints (4 const. modified): original work - our implementation

RESULTS: ORIGINAL WORK VS OUR IMPLEMENTATION WITH THE DUMMY STATE

Trace length	Align. cost	Average cost
1-50	4.34	4.28
51-100	7.1	9.76
101-150	9.81	16.26
151-200	14.4	21.67

10 constraints (6 const. modified): original work - our implementation

Trace length	Align. cost	Average cost
1-50	5.23	6.24
51-100	8.12	9.52
101-150	10.96	14.53
151-200	16.3	20.64
201-250		25.55

15 constraints (6 const. modified): original work - our implementation

RESULTS: ORIGINAL WORK VS OUR IMPLEMENTATION WITH THE DUMMY STATE

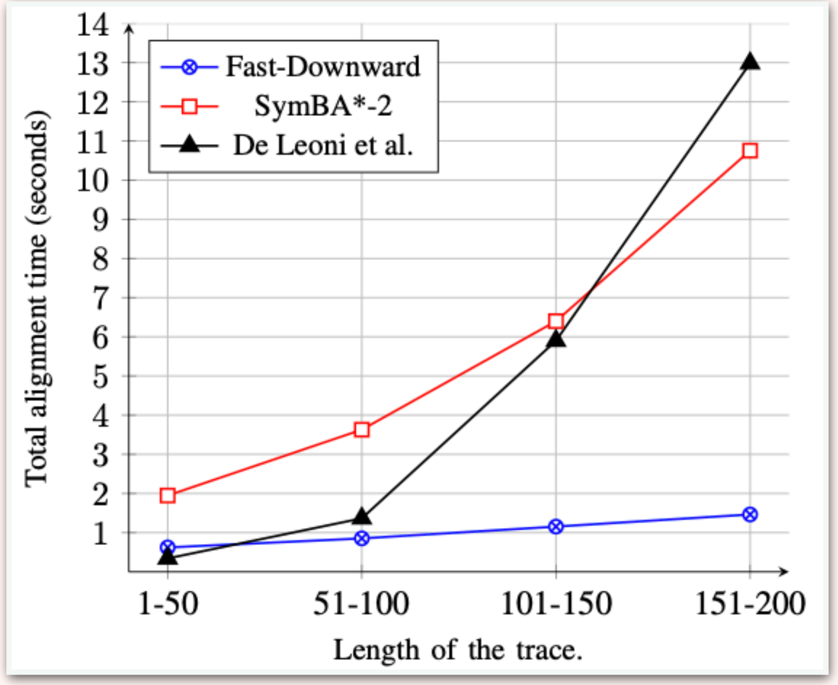
Trace length	Align. cost	Average cost
1-50	3.21	3.8
51-100	6.12	5.95
101-150	10.35	9.51
151-200	14.2	12.34

15 constraints (4 const. modified): original work - our implementation

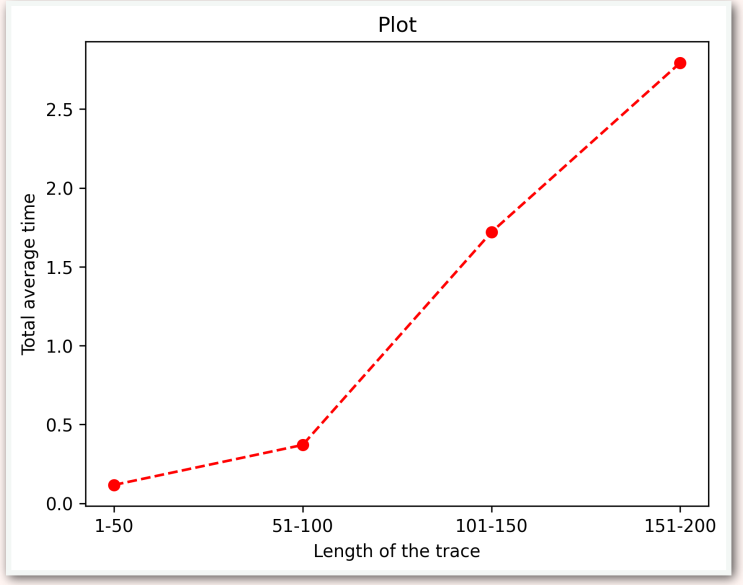
RESULTS: ORIGINAL WORK VS OUR IMPLEMENTATION WITH THE DUMMY STATE

Trace length	Align. time	Average time with dummy state
1-50	0.62	0.12
51-100	0.85	0.37
101-150	1.15	1.72
151-200	1.46	2.79

10 constraints (3 const. modified): original work - our implementation



10 constraints (3 const. modified): original work

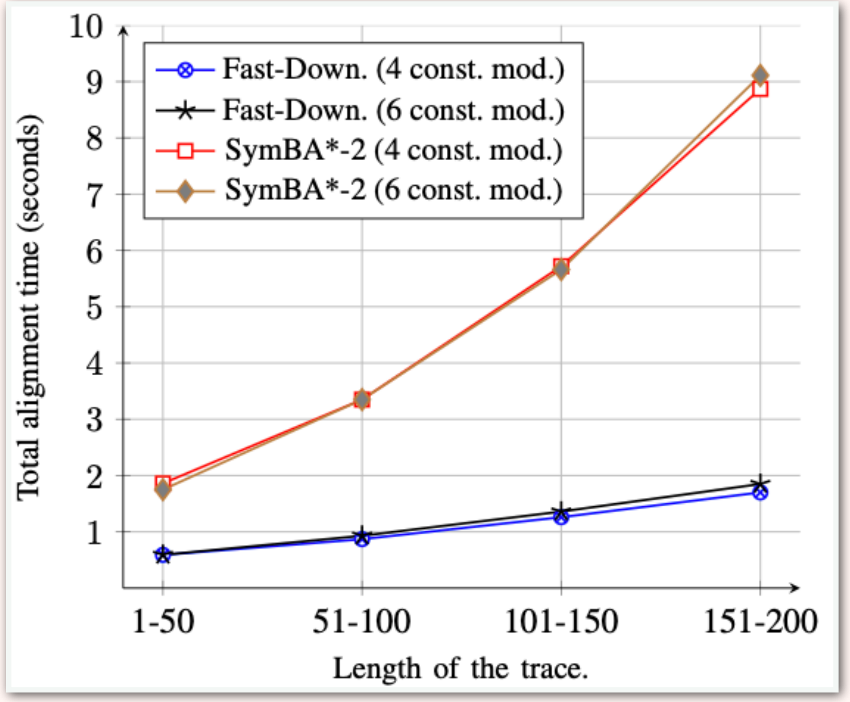


10 constraints (3 const. modified): our implementation

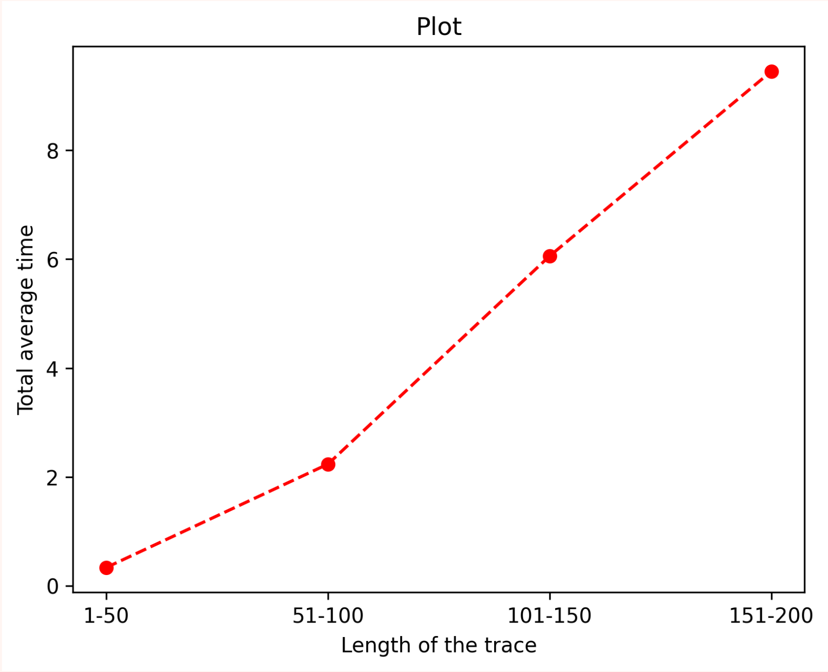
RESULTS: ORIGINAL WORK VS OUR IMPLEMENTATION WITH THE DUMMY STATE

Trace length	Align. time	Average time with dummy state
1-50	0.59	0.33
51-100	0.87	2.23
101-150	1.26	6.06
151-200	1.7	9.45

10 constraints (4 const. modified): original work - our implementation



10 constraints (4 const. modified): original work

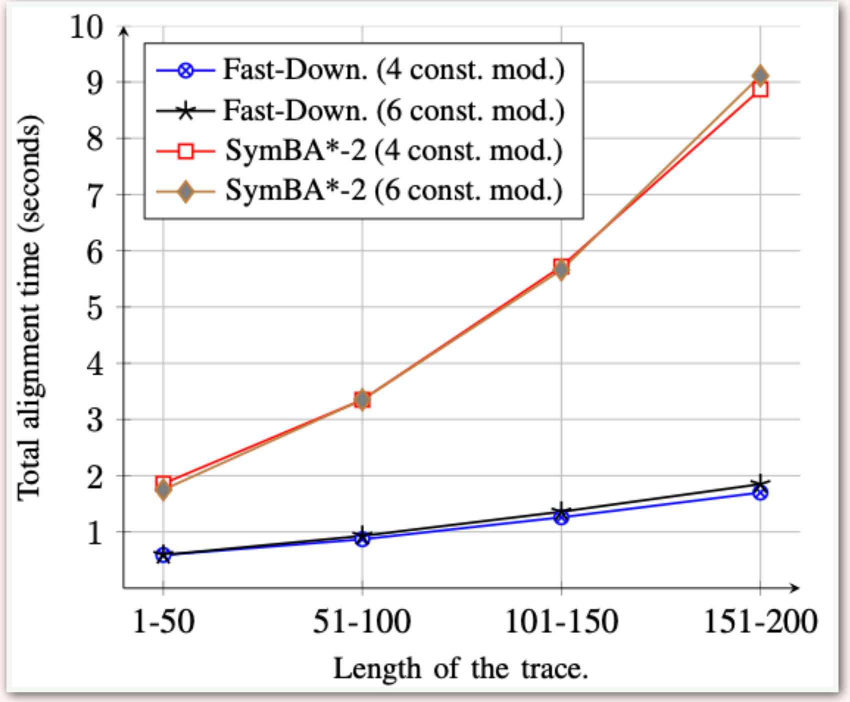


10 constraints (4 const. modified): our implementation

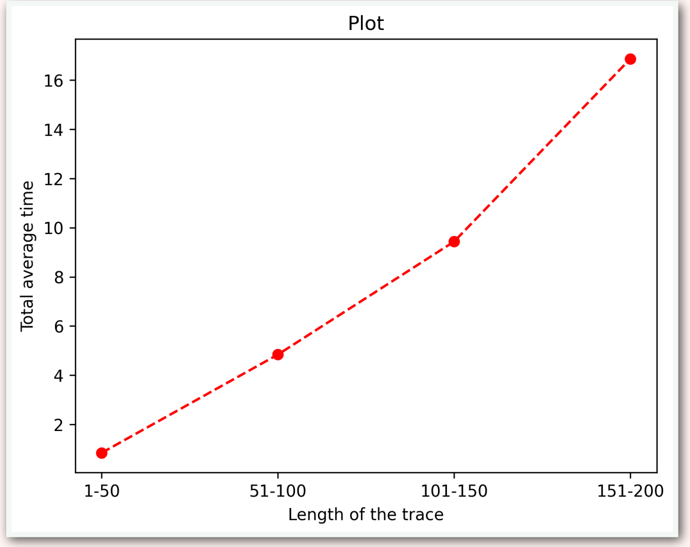
RESULTS: ORIGINAL WORK VS OUR IMPLEMENTATION WITH THE DUMMY STATE

Trace length	Align. time	Average time with dummy state
1-50	0.59	0.84
51-100	0.93	4.84
101-150	1.36	9.44
151-200	1.85	16.87

10 constraints (6 const. modified): original work - our implementation



10 constraints (6 const. modified): original work

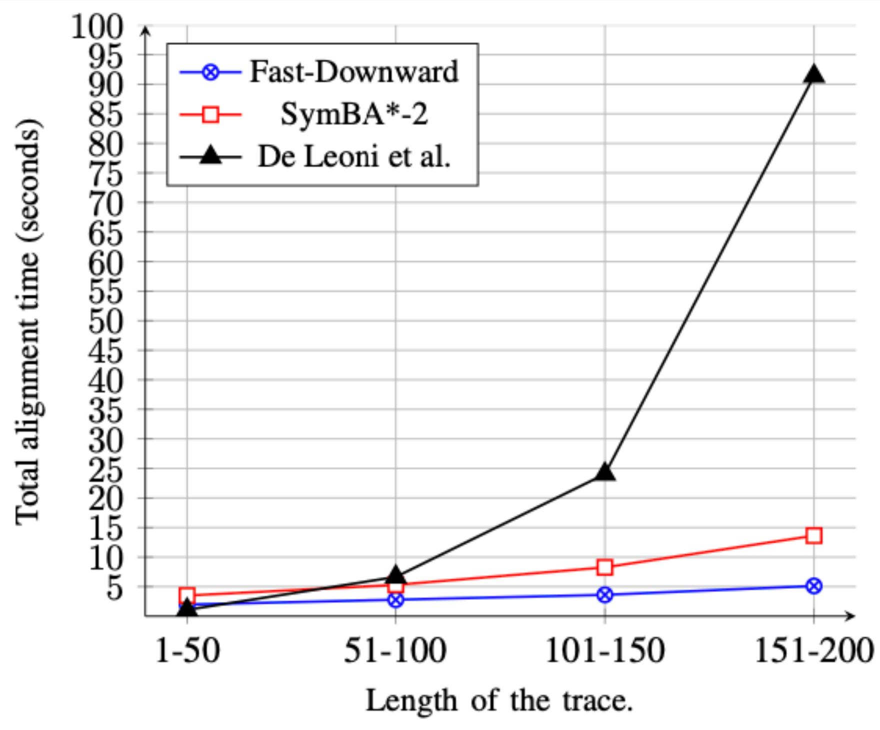


10 constraints (6 const. modified): our implementation

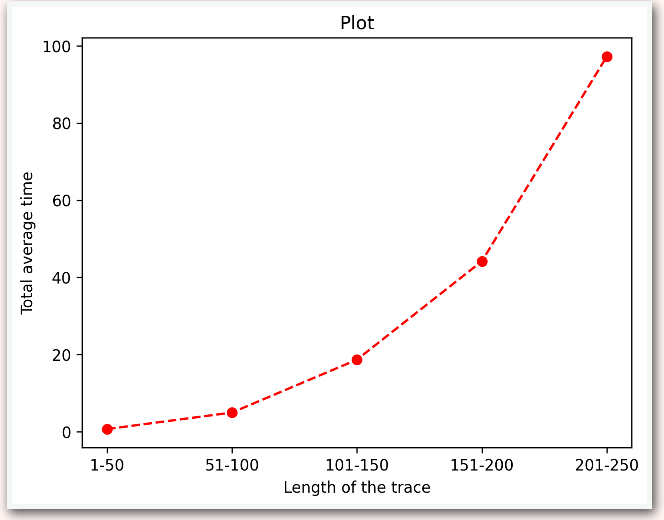
RESULTS: ORIGINAL WORK VS OUR IMPLEMENTATION WITH THE DUMMY STATE

Trace length	Align. time	Average time with dummy state
1-50	1.97	0.69
51-100	2.79	4.97
101-150	3.61	18.68
151-200	5.12	44.18
201-250		97.24

15 constraints (3 const. modified): original work - our implementation



15 constraints (3 const. modified): original work

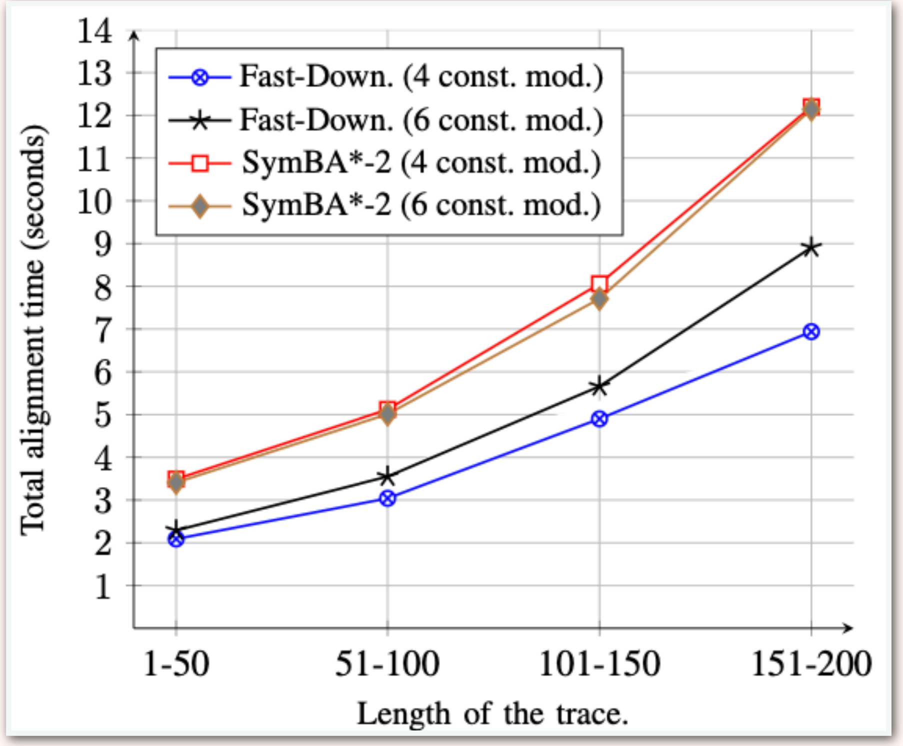


15 constraints (3 const. modified): our implementation

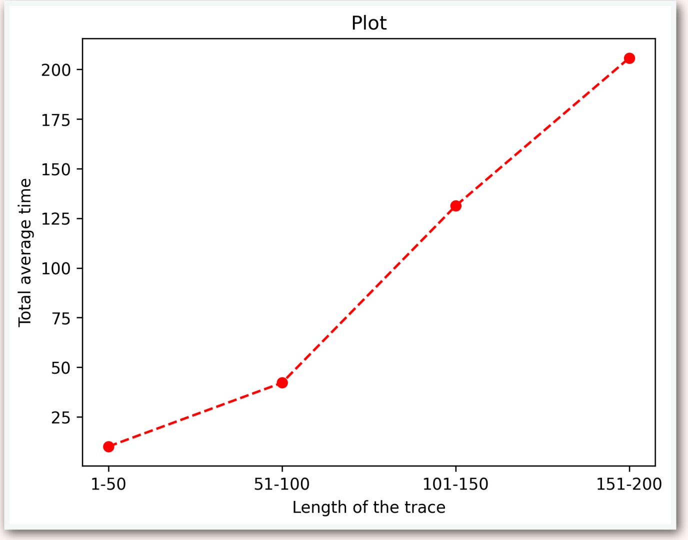
RESULTS: ORIGINAL WORK VS OUR IMPLEMENTATION WITH THE DUMMY STATE

Trace length	Align. time	Average time with dummy state
1-50	2.09	10.11
51-100	3.04	42.18
101-150	4.9	131.29
151-200	6.94	205.73

15 constraints (4 const. modified): original work - our implementation



15 constraints (4 const. modified): original work

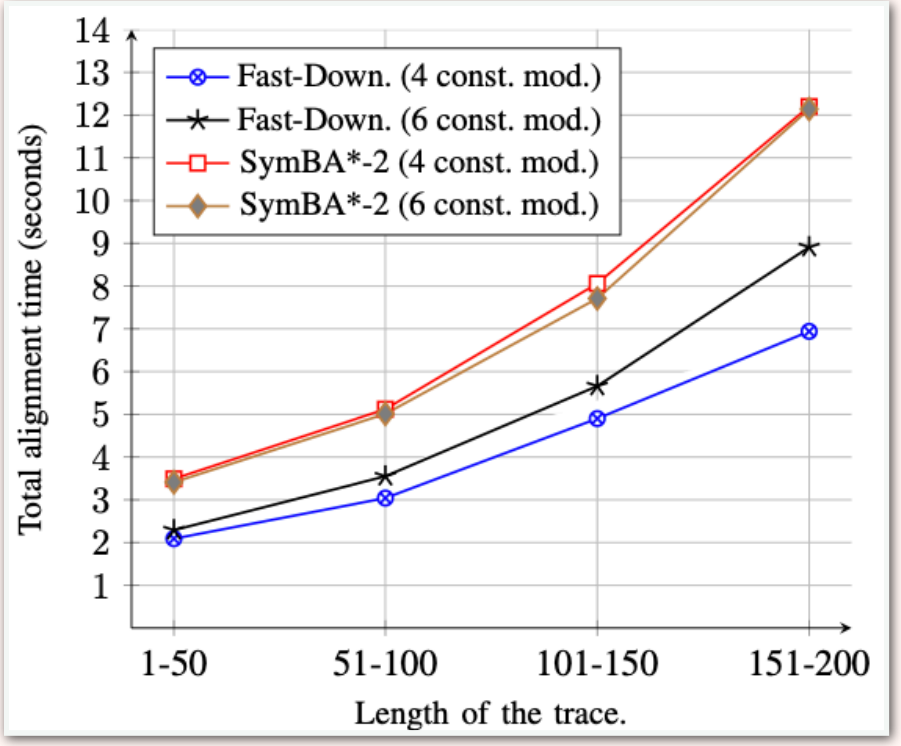


15 constraints (4 const. modified): our implementation

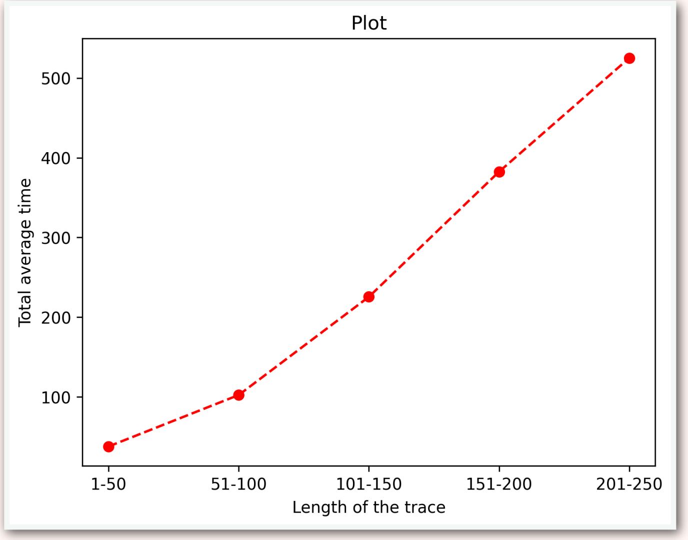
RESULTS: ORIGINAL WORK VS OUR IMPLEMENTATION WITH THE DUMMY STATE

Trace length	Align. time	Average time with dummy state
1-50	2.29	37.87
51-100	3.55	102.43
101-150	5.66	225.57
151-200	8.91	382.51
201-250		525.07

15 constraints (6 const. modified): original work - our implementation



15 constraints (6 const. modified): original work

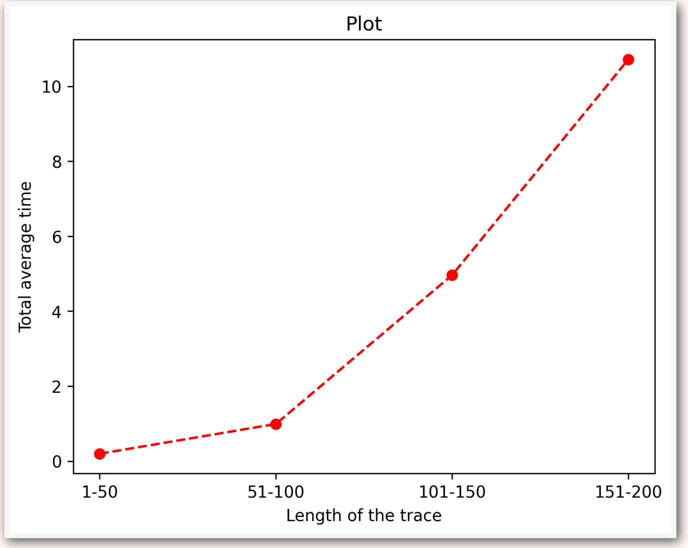


15 constraints (6 const. modified): our implementation

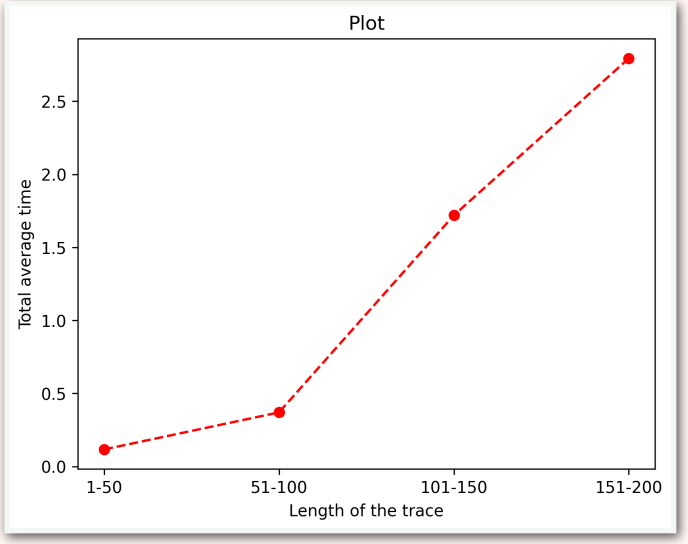
RESULTS: OUR IMPLEMENTATION WITH DUMMY STATE VS OUR IMPLEMENTATION WITHOUT THE DUMMY STATE

Trace length	Average time with dummy state	Average time without dummy state
1-50	0.12	0.20
51-100	0.37	0.99
101-150	1.72	4.96
151-200	2.79	10.72

10 constraints (3 const. modified): our implementations



10 constraints (3 const. modified): without dummy state

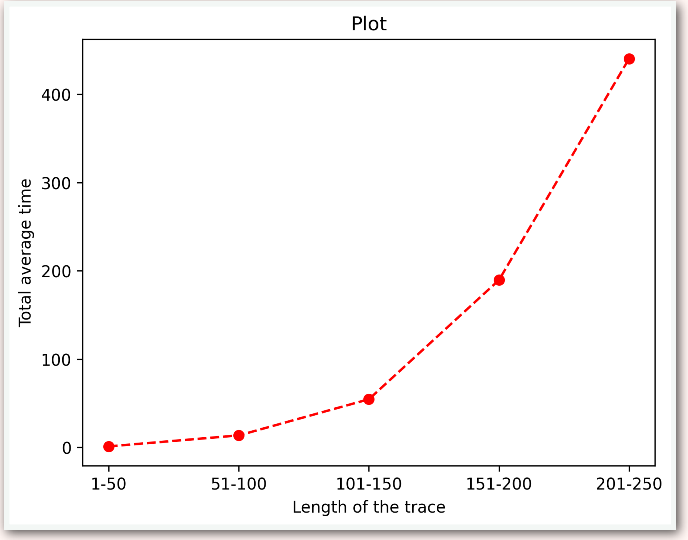


10 constraints (3 const. modified): with dummy state

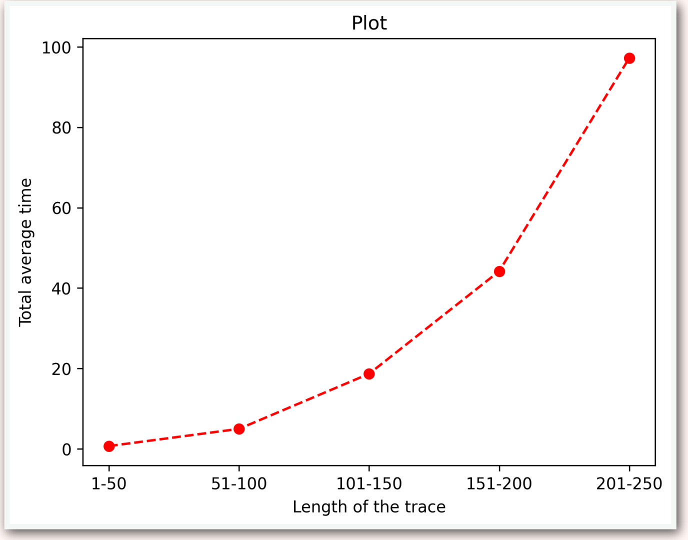
RESULTS: OUR IMPLEMENTATION WITH DUMMY STATE VS OUR IMPLEMENTATION WITHOUT THE DUMMY STATE

Trace length	Average time with dummy state	Average time without dummy state
1-50	0.69	1.14
51-100	4.97	13.58
101-150	18.68	54.54
151-200	44.18	189.59
201-250	97.24	440.37

15 constraints (3 const. modified): original work - our implementations



15 constraints (3 const. modified): without dummy state



15 constraints (3 const. modified): with dummy state

RESULTS: OUR IMPLEMENTATION WITH DUMMY STATE VS OUR IMPLEMENTATION WITHOUT THE DUMMY STATE

Trace length	Average cost with dummy state	Average cost without dummy state
1-50	1.79	1.79
51-100	2.13	2.13
101-150	3.04	3.04
151-200	3.81	3.81

10 constraints (3 const. modified): our implementations

Trace length	Average cost with dummy state	Average cost without dummy state
1-50	1.73	1.73
51-100	2.26	2.26
101-150	3.09	3.09
151-200	4.24	4.24
201-250	5.33	5.33

15 constraints (3 const. modified): our implementations

CONCLUSIONS

- **Original works VS Our implementation with the dummy state:**
  - **Good costs**
  - **Bad computational time**
    - **Different device for the experiments**
    - **Different version of the planning system Fast Downward**
- **Our implementations:**
  - **Equal cost**
  - **Implementation with the dummy state is faster**

THANK YOU FOR YOUR ATTENTION!

REFERENCES

- **Giuseppe De Giacomo, Fabrizio Maria Maggi, Andrea Marrella, and Fabio Patrizi. On the disruptive effectiveness of automated planning for LTL$_f$- based trace alignment. In** *Proceedings of the AAAI Conference on Artificial Intelligence,* **volume 31, 2017.**
- **Giuseppe De Giacomo, Fabrizio Maria Maggi, Andrea Marrella, and Sebastian Sardina. Computing trace alignment against declarative process models through planning. In** *Twenty-Sixth International Conference on Automated Planning and Scheduling,* **2016.**
- **Giuseppe De Giacomo and Moshe Y Vardi. Linear temporal logic and linear dynamic logic on finite traces. In** *Twenty-Third International Joint Conference on Artificial Intelligence,* **2013.**
- **Massimiliano De Leoni, Fabrizio M Maggi, and Wil MP van der Aalst. An alignment-based framework to check the conformance of declarative process models and to preprocess event-log data.** *Information Systems,* **47:258–277, 2015.**
- **Massimiliano De Leoni, Fabrizio Maria Maggi, and Wil MP van der Aalst. Aligning event logs and declarative process models for conformance checking. In** *International Conference on Business Process Management,* **pages 82–97. Springer, 2012.**
- **Wil MP Van Der Aalst, Marcello La Rosa, and Flávia Maria Santoro. Business process management, 2016.**
- **Giuseppe De Giacomo and Moshe Vardi. Synthesis for ltl and ldl on finite traces. In Twenty-Fourth International Joint Conference on Artificial Intelligence, 2015.**
- **Claudio Di Ciccio, Mario Luca Bernardi, Marta Cimitile, and Fabrizio Maria Maggi. Generating event logs through the simulation of declare models. In Workshop on Enter- prise and Organizational Modeling and Simulation, pages 20–36. Springer, 2015.**