SAPIENZA
UNIVERSITÀ DI ROMA

Table of Contents

SAPIENZA
UNIVERSITÀ DI ROMA

SAPIENZA
UNIVERSITÀ DI ROMA

Business Process

- BP defines the temporal (partial) ordering of some activities
- Activities execution are stored in logs → set of traces
- Sometimes the log traces can be *inconsistent* with the expected process behavior → need to repair the traces
- Process Mining is the analysis of (business) process starting from event logs

SAPIENZA
UNIVERSITÀ DI ROMA

Trace Alingment for Business Process

Trace alignment is the problem of:

- Checking whether an actual trace conforms to the expected process behavior (process model) and, if not...
- Finding a “minimal set” of changes that “align” the trace to the process
  - Changes consist in adding / removing activities from traces
- Focus on trace alignment against *declarative* specifications (descriptive) → constraints expressed in DECLARE language or LTL*f*

SAPIENZA
UNIVERSITÀ DI ROMA

Trace Alingment in Declarative BP

Given:

- A trace ρ over a finite event (possible activity) alphabet $\Sigma = \{\sigma 1, \ldots, \sigma n\}$
- A constraint φ
- A *cost* function associating «positive» cost to additions / deletions of events

Find a trace ρ' such that:

- ρ' satisfies φ, i.e. $\rho' \models \varphi$
- cost(ρ, ρ') is minimal (turning from the original trace to the other)

Trace Alingment in Declarative BP – LTL*f*

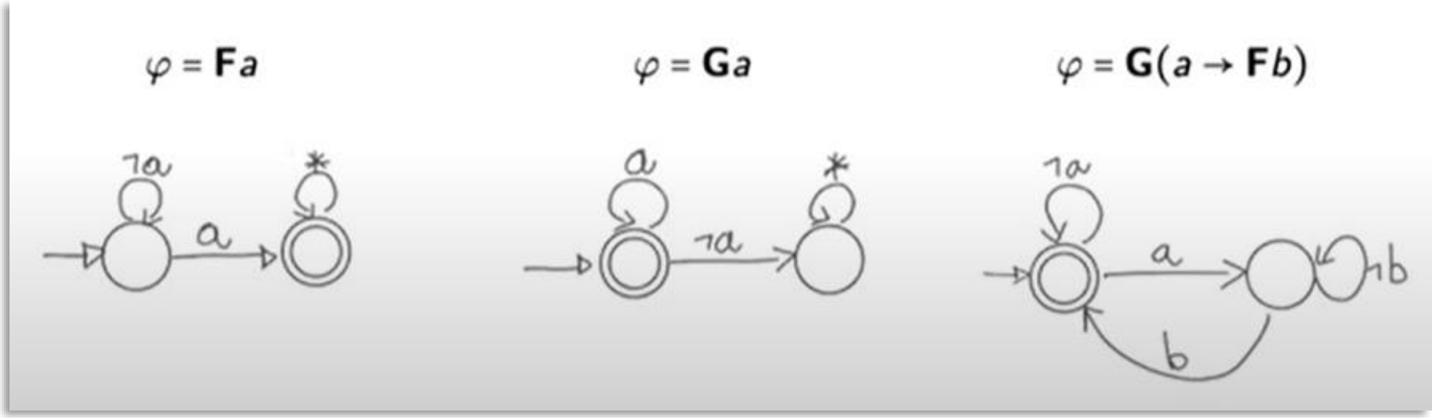
Constraints specification can be done through DECLARE language template that can be translated in LTL*f* translation

DECLARE template	LTL*f* translation
Existence(a)	**F***a*
Response(a, b)	**G**(*a* → **F**b)
Choice(a, b)	**F***a* V **F***b*
ChainResponse(a, b)	**G**(*a* → **X***b*)

SAPIENZA
UNIVERSITÀ DI ROMA

LTL*f* Formulas as NFAs (property)

- Every LTLf formula has a corresponding (exponential) NFA $A\varphi$ s.t
  - For every trace $\rho : \rho \models \varphi \leftrightarrow A\varphi$ *accepts* ρ



Automata-based Solution for Trace Alignment

Given:

- trace ρ
- constraint φ

Define two automatons:

- *Augmented trace automaton* T^+ : accepts all modifications of ρ (where the changes are marked)
- *Augmented constraint automaton* A^+ : accepts all traces that satisfy φ && all modifications of ρ that satisfy φ

Find minimal-cost ρ' s.t. is accepted by T^+ && A^+ (satisfies both)

- Corresponds to find minimal-cost accepting path ρ' on product automaton $T^+ \times A^+$

Trace Alingment as Planning Problem

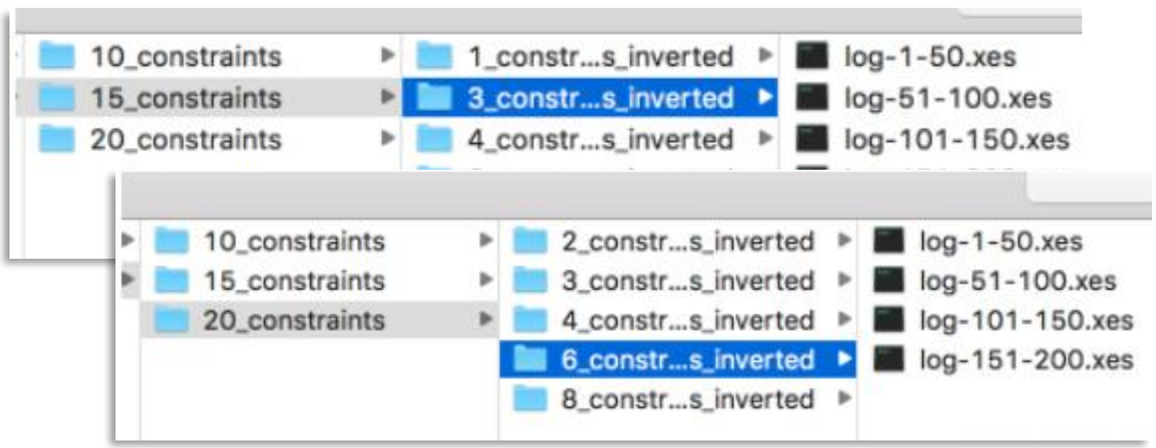
Use Planning to search for minimal-cost *ρ'*

- Domain
  - Models product automaton $T^{+} \times A^{+}$
  - *add* and *del* actions with positive cost (changes of input trace)
  - *sync* actions with null cost model events
- Problem
  - Initial State: all automata in their starting state
  - Goal: all automata in a final state
- Solution
  - Minimal-cost goal-reaching sequence of actions

Implementation – Dataset

Synthetic Logs:

- Traces having the same alphabet of activities, containing 10 / 15 / 20 constraints
- Each log contains different amount of noise (i.e. N constraints are inverted over the total of 10 / 15 / 20 constraints on which the traces are modeled)
- Different lengths:
  - 1-50
  - 51-100
  - 101-150
  - 151-200
  - 201-250



SAPIENZA
UNIVERSITÀ DI ROMA

SAPIENZA
UNIVERSITÀ DI ROMA

Implementation – from Traces/Constraints to Automata

Given a trace $t = e_1 \ldots e_m$, build a Trace **Automaton T+** (DFA)

- $T = (\Sigma_t , Q_t , q^t_0 , \rho_t , F_t)$
- Σ_t = set of all the events occurring in t
- Q_t = set of states
- q^t_0 = initial state
- ρ_t = transition function
- F_t = final state (for Trace Automaton is always one)

Build a **Constraint Automaton A+** (NFA)

- $A = (\Sigma_a , Q_a , q^a_0 , \rho_a , F_a)$
- Σ_a = alphabet
- Q_a = set of states
- q^a_0 = initial state
- ρ_a = transition function
- F_a = set of final state

***** Each propositional interpretation in a trace cornensponds to one proposition (a singleton) – in process mining only one event True at each time, in LTLf can be more than one event True at each time

LTLf2DFA and MONA representation

G(activity14_complete ->**X**(activity15_complete))

Transitions:
State 0**: XX** -> state 1
State 1: 0**X** -> state 1
State 1: 1**X** -> state 2
State 2: 00 -> state 3
State 2: 01 -> state 1
State 2: 10 -> state 3
State 2: 11 -> state 2
State 3: **XX** -> state 3

Expanded state transitions…
State 0: 00 -> state 1
State 0: 01 -> state 1
State 0: 10 -> state 1
State 0: 11 -> state 1
State 1: 00 -> state 1
State 1: 01 -> state 1
State 1: 10 -> state 2
State 1: 11 -> state 2
State 2: 00 -> state 3
State 2: 01 -> state 1
State 2: 10 -> state 3
State 2: 11 -> state 2
State 3: 00 -> state 3
State 3: 01 -> state 3
State 3: 10 -> state 3
State 3: 11 -> state 3

SAPIENZA
UNIVERSITÀ DI ROMA

LTLf2DFA and MONA representation

Sum of the labels to have always a singleton :

- If sum > 1 discard the transition
- If sum = 1 take into account the transition with the event True
- If sum = 0 save the negated transition to add in the pddl problem the complemented events present both in the constraints and in the trace
  - We have State 1: 0 -> state 2 , in constraints the event A and in the trace the events A, B, C. We add into the pddl the transitions "s1 B s2" and "s1 C s2"

Transitions:

State 0: 00 -> state 1
State 0: 01 -> state 1
State 0: 10 -> state 1
State 0: 11 -> state 1
State 1: 00 -> state 1
State 1: 01 -> state 1
State 1: 10 -> state 2
State 1: 11 -> state 2
State 2: 00 -> state 3
State 2: 01 -> state 1
State 2: 10 -> state 3
State 2: 11 -> state 2
State 3: 00 -> state 3
State 3: 01 -> state 3
State 3: 10 -> state 3
State 3: 11 -> state 3

Implementation – PDDL

Domain

Actions
- add, del, sync (as defined in the original paper)

```
(:action add
 :parameters (?e - activity)
 :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)))
```

OBJECTS
State and Activity instances involved in the trace automaton and in any constraint automaton

INIT
Transitions that connect two different states
Curr state + Final state of all automaton

GOAL
conjunction of the accepting states of the trace automaton and of the constraint automata

METRIC
(:metric **minimize** (total-cost))

Problem

SAPIENZA
UNIVERSITÀ DI ROMA

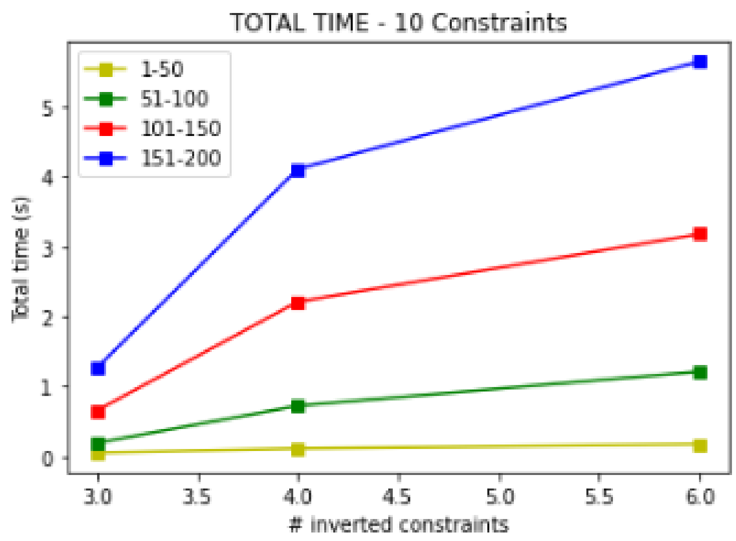
Results – 10 constraints

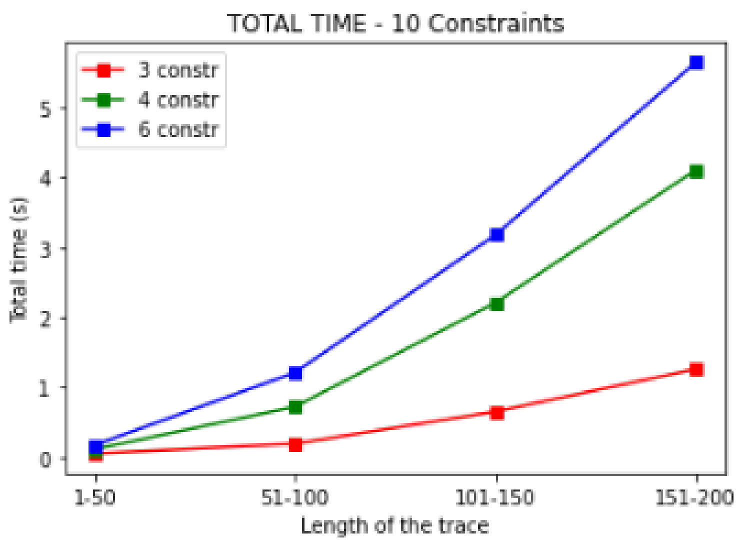
Trace length	3 inverted index		4 inverted index		6 inverted index	
	Alignment Cost	Total time	Alignment Cost	Total time	Alignment Cost	Total time
1-50	1.77	0.05	2.74	0.11	4.23	0.17
51-100	2.11	0.19	5.86	0.72	9.74	1.21
101-150	3.03	0.65	9.68	2.21	16.23	3.18
151-200	3.79	1.26	13.4	4.11	21.63	5.65

- **Fast-downward planner – h_{max} heuristic**
- **Alignment Cost – number of add/del to repair the trace (avg on 100 traces)**
- **Total Time in seconds (avg on 100 traces)**

SAPIENZA
UNIVERSITÀ DI ROMA

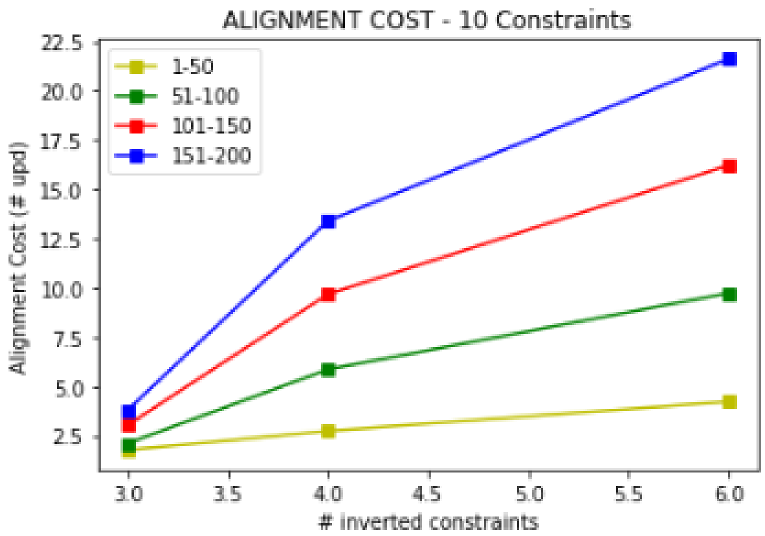
Results – 10 constraints

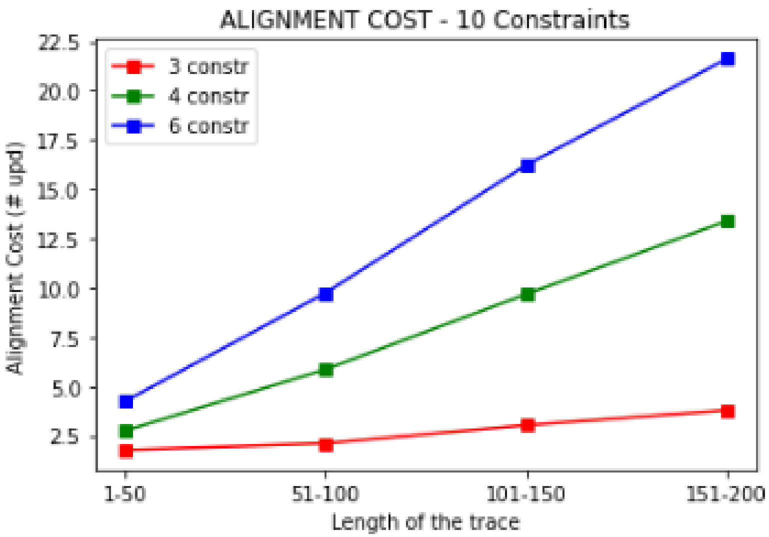




- **Fast-downward planner – h_{max} heuristic**
- **Alignment Cost – number of add/del to repair the trace (avg on 100 traces)**
- **Total Time in seconds (avg on 100 traces)**

Results – 10 constraints





- **Fast-downward planner – h_{max} heuristic**
- **Alignment Cost – number of add/del to repair the trace (avg on 100 traces)**
- **Total Time in seconds (avg on 100 traces)**

SAPIENZA
UNIVERSITÀ DI ROMA

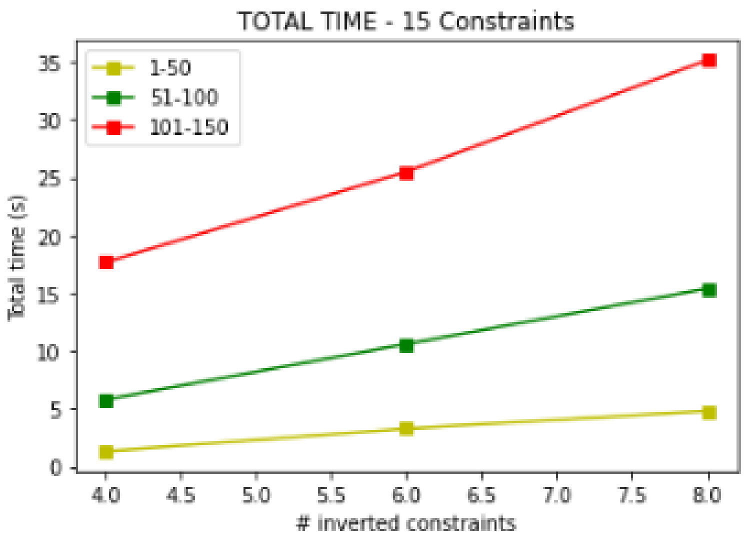
Results – 15 constraints

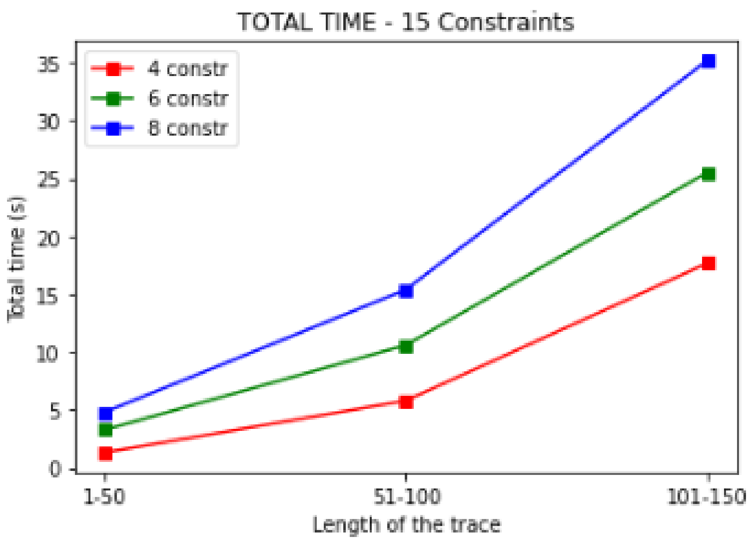
Trace length	4 inverted index		6 inverted index		8 inverted index	
	Alignment Cost	Total time	Alignment Cost	Total time	Alignment Cost	Total time
1-50	3.8	1.3	6.24	3.26	7.64	4.79
51-100	5.94	5.78	9.52	10.59	13.09	15.39
101-150	9.49	17.66	14.47	25.5	19.49	35.17

- **Fast-downward planner – h_{max} heuristic**
- **Alignment Cost – number of add/del to repair the trace (avg on 100 traces)**
- **Total Time in seconds (avg on 100 traces)**

SAPIENZA
UNIVERSITÀ DI ROMA

Results – 15 constraints

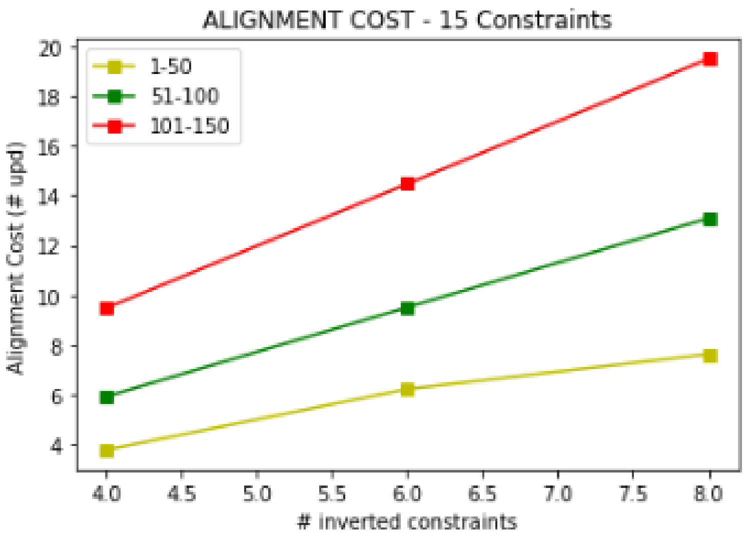


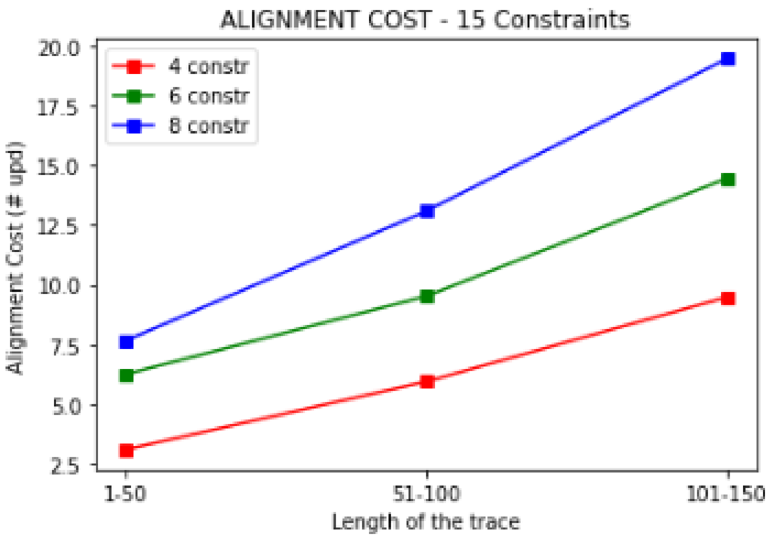


- **Fast-downward planner – h_{max} heuristic**
- **Alignment Cost – number of add/del to repair the trace (avg on 100 traces)**
- **Total Time in seconds (avg on 100 traces)**

SAPIENZA
UNIVERSITÀ DI ROMA

Results – 15 constraints

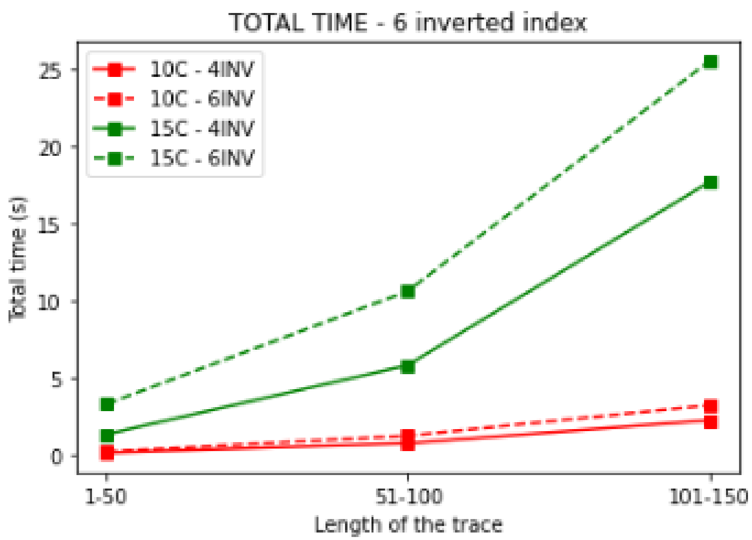


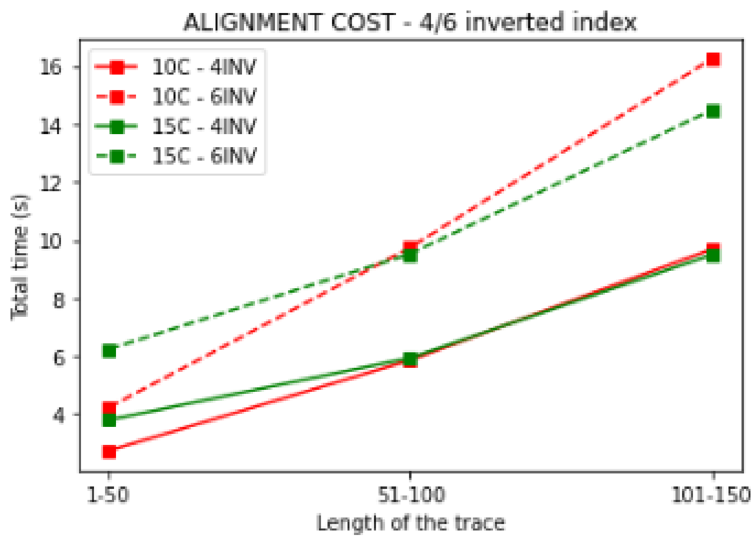


- Fast-downward planner – h_{max} heuristic
- Alignment Cost – number of add/del to repair the trace (avg on 100 traces)
- Total Time in seconds (avg on 100 traces)

SAPIENZA
UNIVERSITÀ DI ROMA

Results – 10 vs 15 constraints on 4/6 inverted





- **Fast-downward planner – h_{max} heuristic**
- **Alignment Cost – number of add/del to repair the trace (avg on 100 traces)**
- **Total Time in seconds (avg on 100 traces)**

SAPIENZA
UNIVERSITÀ DI ROMA

References

On the Disruptive Effectiveness of Automated Planning for LTLf-Based Trace Alignment *Giuseppe De Giacomo, Fabrizio Maria Maggi, Andrea Marrella, Fabio Patrizi*

LTLf2DFA Tool

MONA Tool

Fast Downward Planner

SAPIENZA
UNIVERSITÀ DI ROMA

THANKS

SAPIENZA
UNIVERSITÀ DI ROMA