An algorithm for learning real-time automata

Sicco Verwer, Mathijs de Weerdt, and Cees Witteveen

TU Delft

Delft University of Technology

Overview

- Detecting driving behavior
- Automata
- Learning an automaton
- Real-time automata
- An algorithm for learning real-time automata
- Results
- Conclusions

Truck driving behavior

- Required is a system that detects driving behavior from sensor data
- This system will be used to give real-time feedback to the driver
- However, there is not enough expert knowledge to construct one from
- It is possible to gather loads of sensor data from trucks

An automaton model for driving behavior

- It is beneficial if the system determines the behavior using discrete events:
  - Discrete events are easy to interpret
- A finite state automaton is used:
  - An automaton is interpretable and powerful
- The intuitive idea: a good driver speaks a 'language' different from the 'language' of a bad driver

Automata

- A finite set of states
- A finite set of symbols
- A finite set of state transitions, each labeled with an symbol
- A Boolean output value

- Used to determine whether a string (of discrete events) is an element of a regular language

Learning an automaton

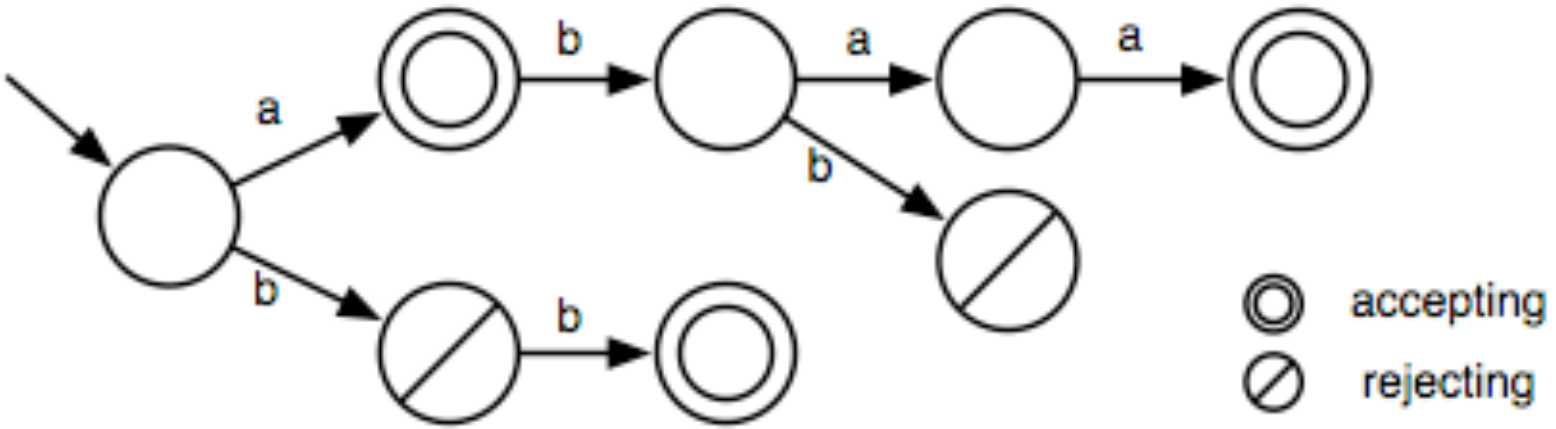
- The input is a finite input sample *S*:
  - { (true, abab), (false, aaabab), .. }

- The output is an automaton such that:
  - It is consistent with *S*
  - It has the least number of states among all possible automata consistent with *S*

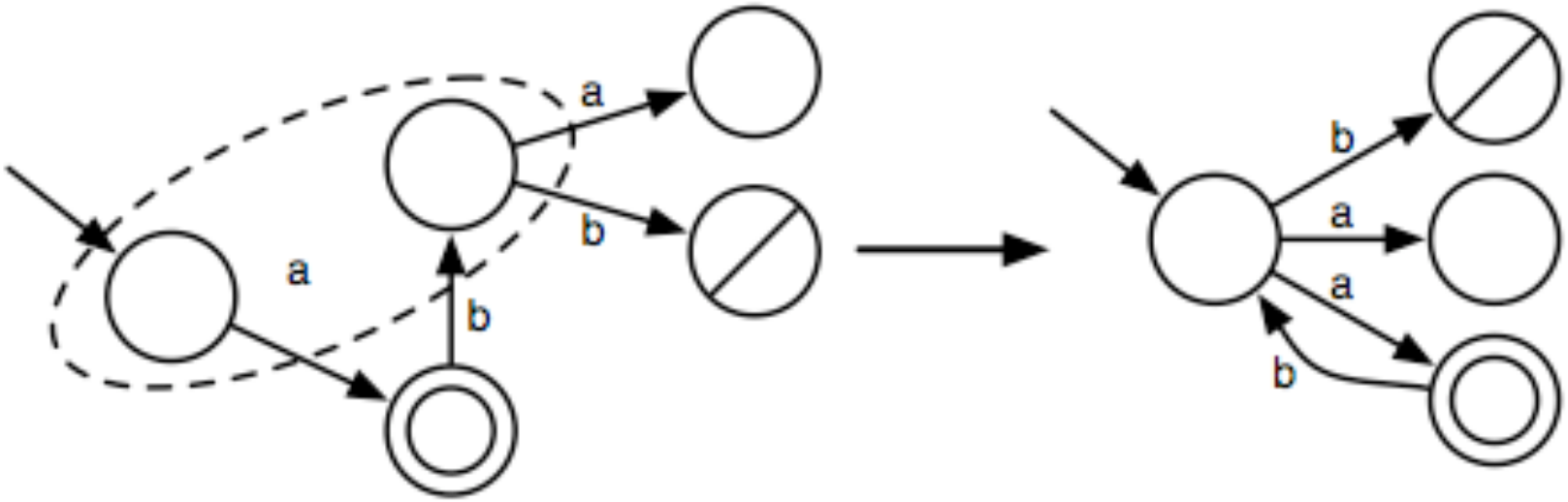
State merging

- Construct an augmented prefix tree acceptor:
  - a tree automaton accepting only the positive examples from the input sample and rejecting the negative
- Merge states of the automaton into one until no more merges are possible:
  - Two states *q* and *q'* can be merged if the data at *q* is consistent with the data at *q'*
- Optionally backtrack or make use some other search mechanism

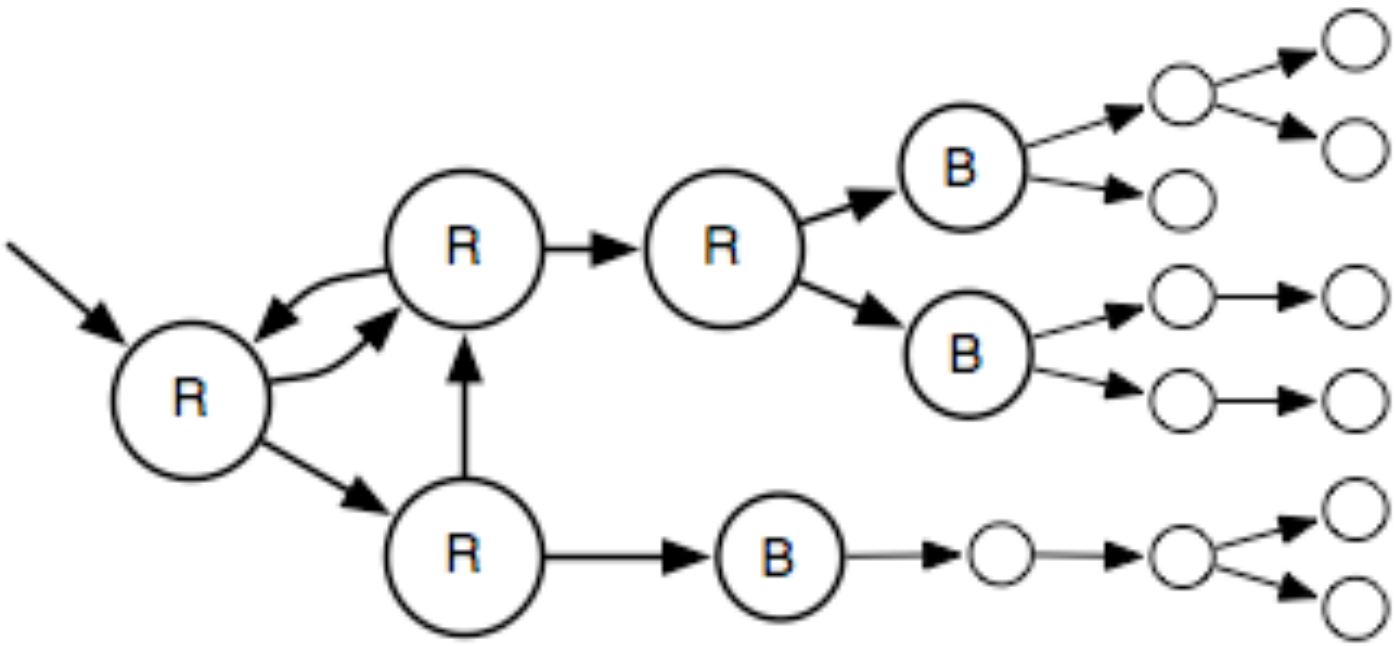
Augmented prefix tree acceptor



Merging a state



Red blue framework



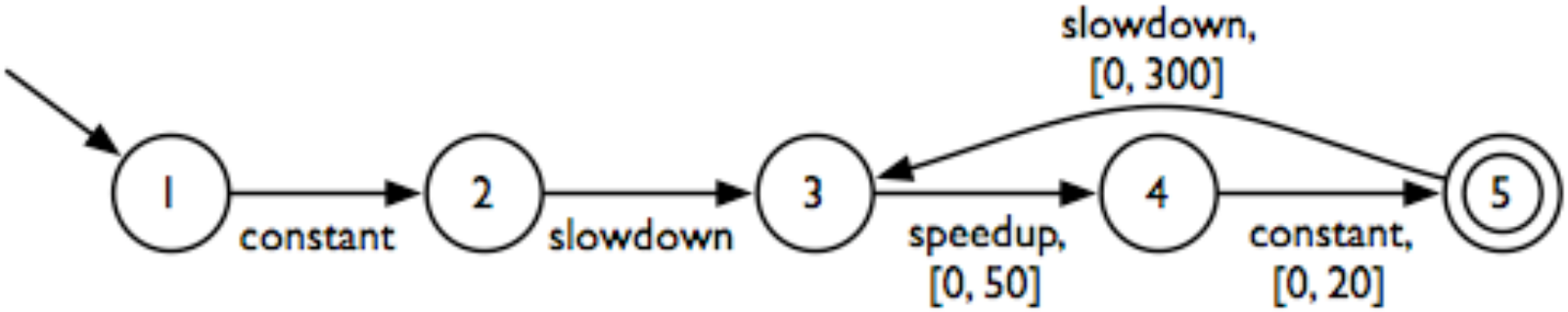
Evidence driven state merging

- Evidence driven state merging currently is a well-known algorithm for identifying DFAs
- A merge is performed if:
  - It is consistent
  - It has highest score amongst all possible merges
- The evidence score is:
  - # positive states merged with positive states + # negative states merged with negative states
- No backtracking is performed

Real-time automata

- A state transition can depend on the time delay *d* between two consecutive events:
  - state transitions optionally contain a guard: $d \in [t,t']$
  - a transition can fire only if its guard is satisfied
- In normal timed automata there can be a guard between any two events

Harmonica behavior



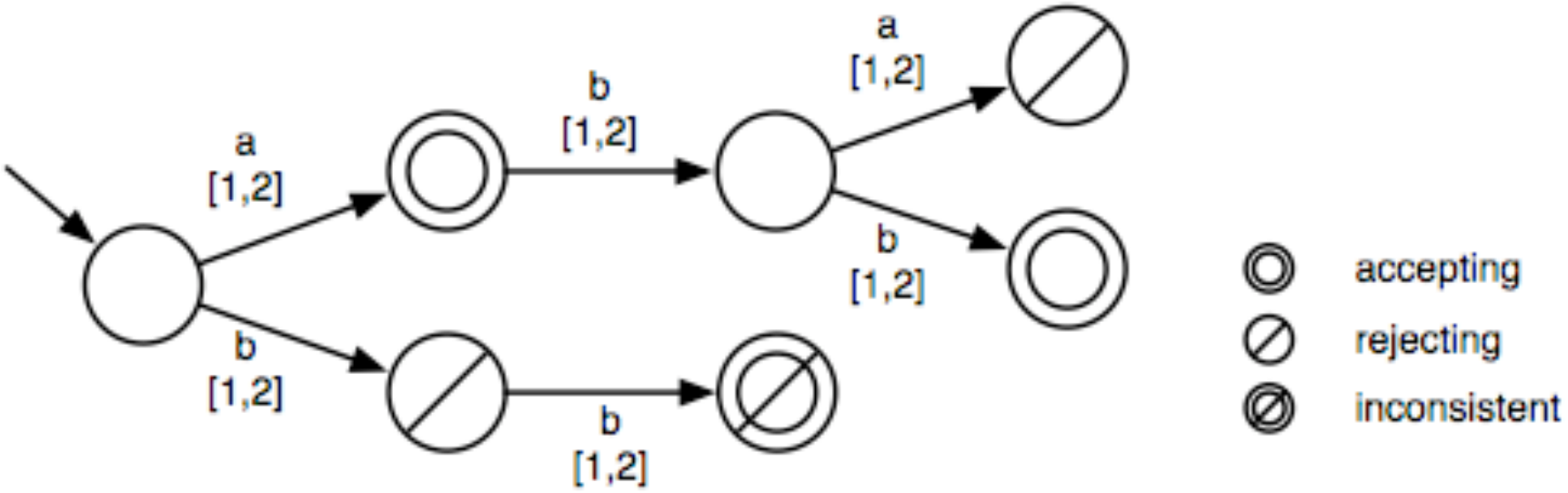
Learning a real-time automaton

- The input is a finite timed input sample *S*:
  - {(true, (a,1.0)(b,3.4)..),(false, (a,0.2)(a,0.5)..).. }
- The output is a real-time automaton such that:
  - It is consistent with *S*
  - It has the least number of transitions among all possible automata consistent with *S*

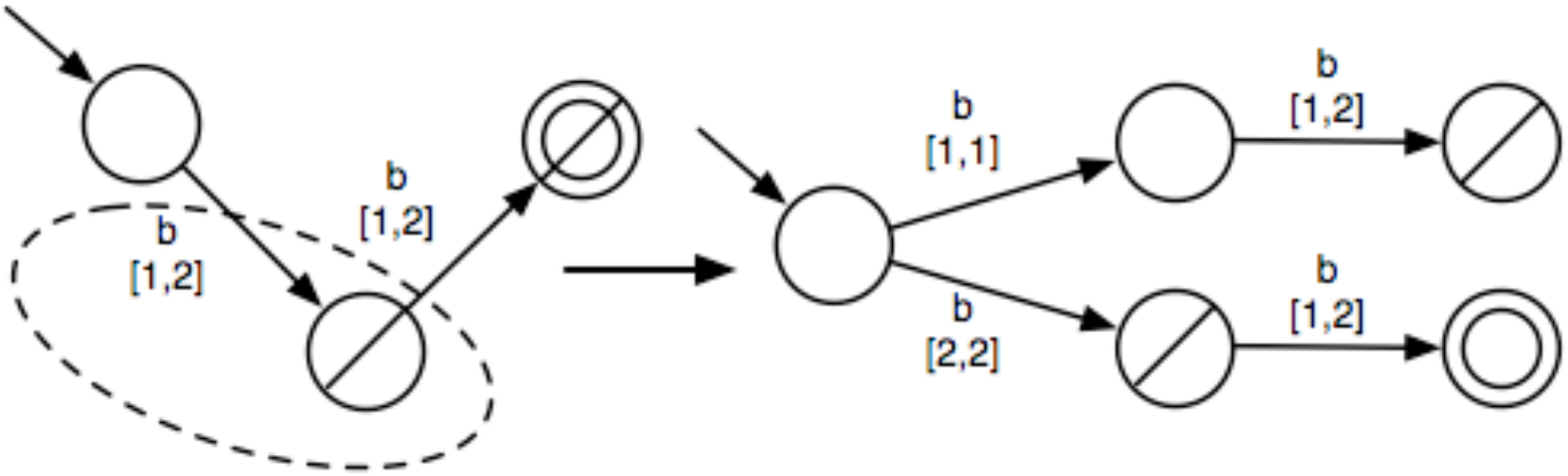
Learning a real-time automaton

- Construct a timed augmented prefix tree acceptor
- Merge states of the automaton
- Split transitions of the automaton into two:
  - $[t, t'] \rightarrow [t,t''], [t'' + l, t']$
- Optionally backtrack or make use some other search mechanism

Prefix tree delay automaton



Splitting a transition



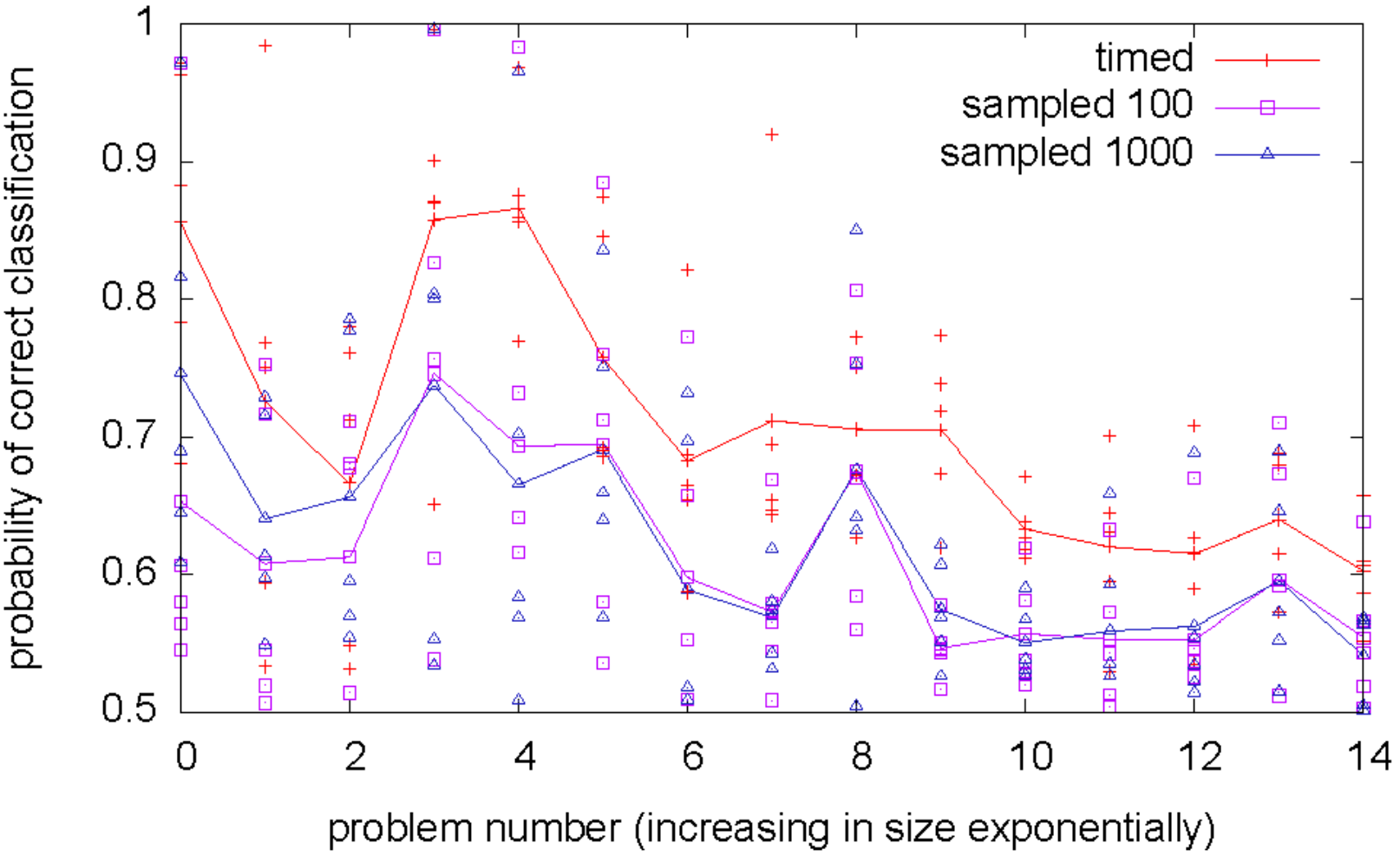
Timed evidence

- A tail is the suffix of an example starting at a blue node:
  - *(a,1)(b,2)(a,2)...(a,1)(a,3)(b,5)*
- The probability that two tails end up in the same state is determined by how 'close' their time values are
- For each tail we divide the EDSM score by the distance from its closest tail

Experiments

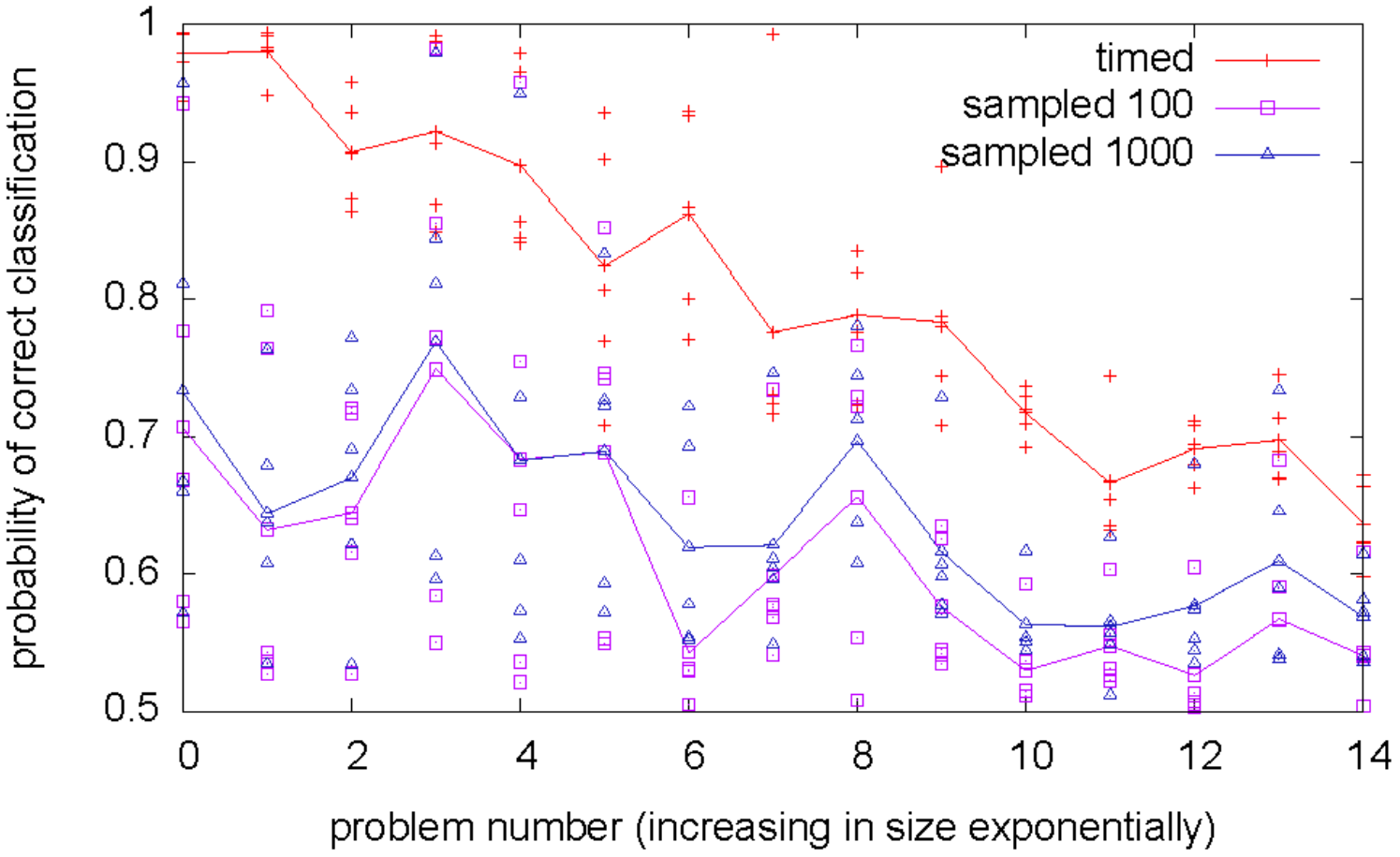
- Data is generated randomly from randomly generated real-time automata with:
  - 2, 4, 8, 16, 32 states
  - 1/2, 1, and 2 times #states of split points
  - 10000 different possibly time values
- Inputs: 50, 500, 1000, 2000, 5000, and 10000 samples
- We compared with a red blue state merging algorithm on the same data, sampled at a fixed rate: *(a,300)* → *aaa*

Results - 50 samples



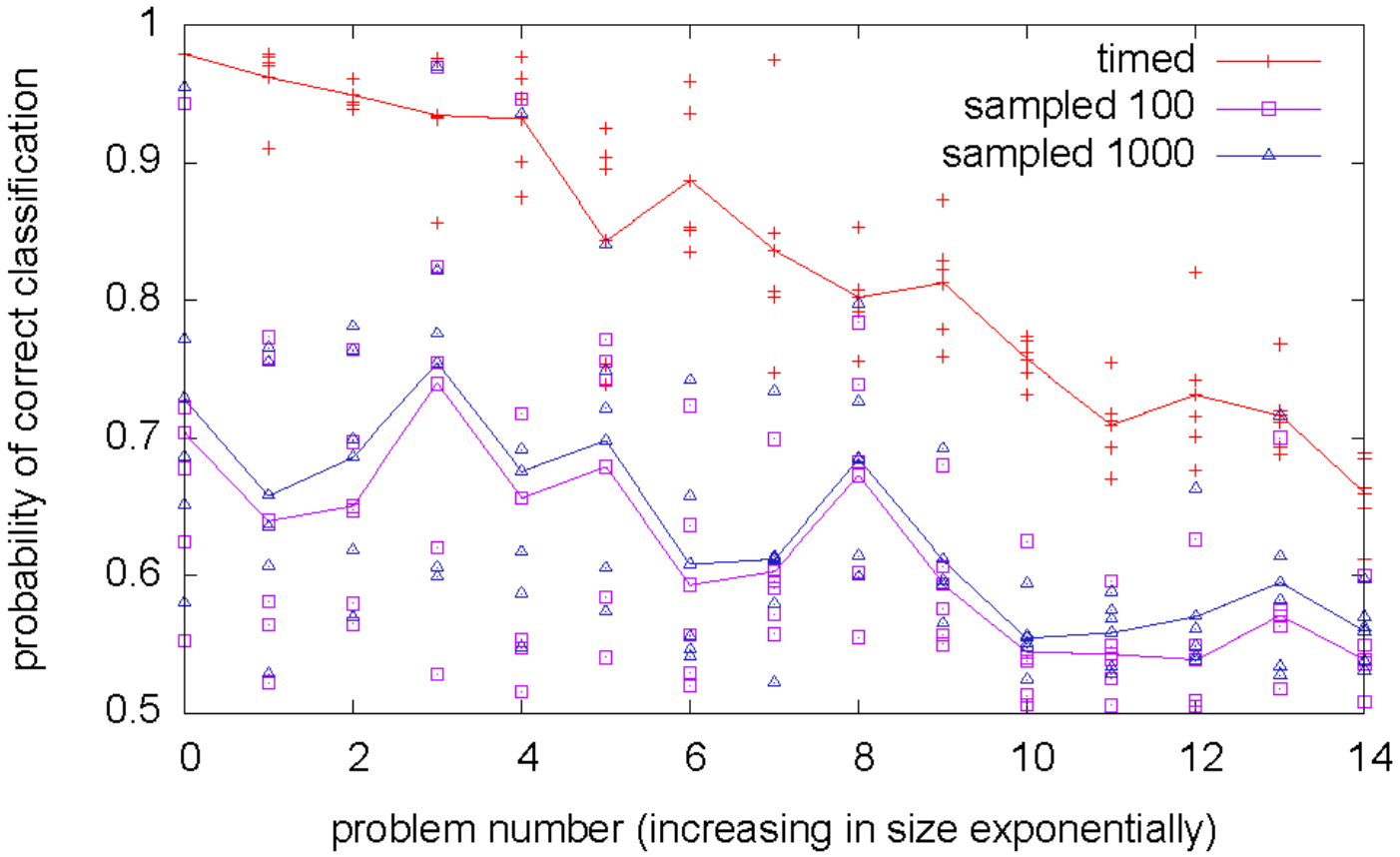
TU Delft

Results - 500 samples

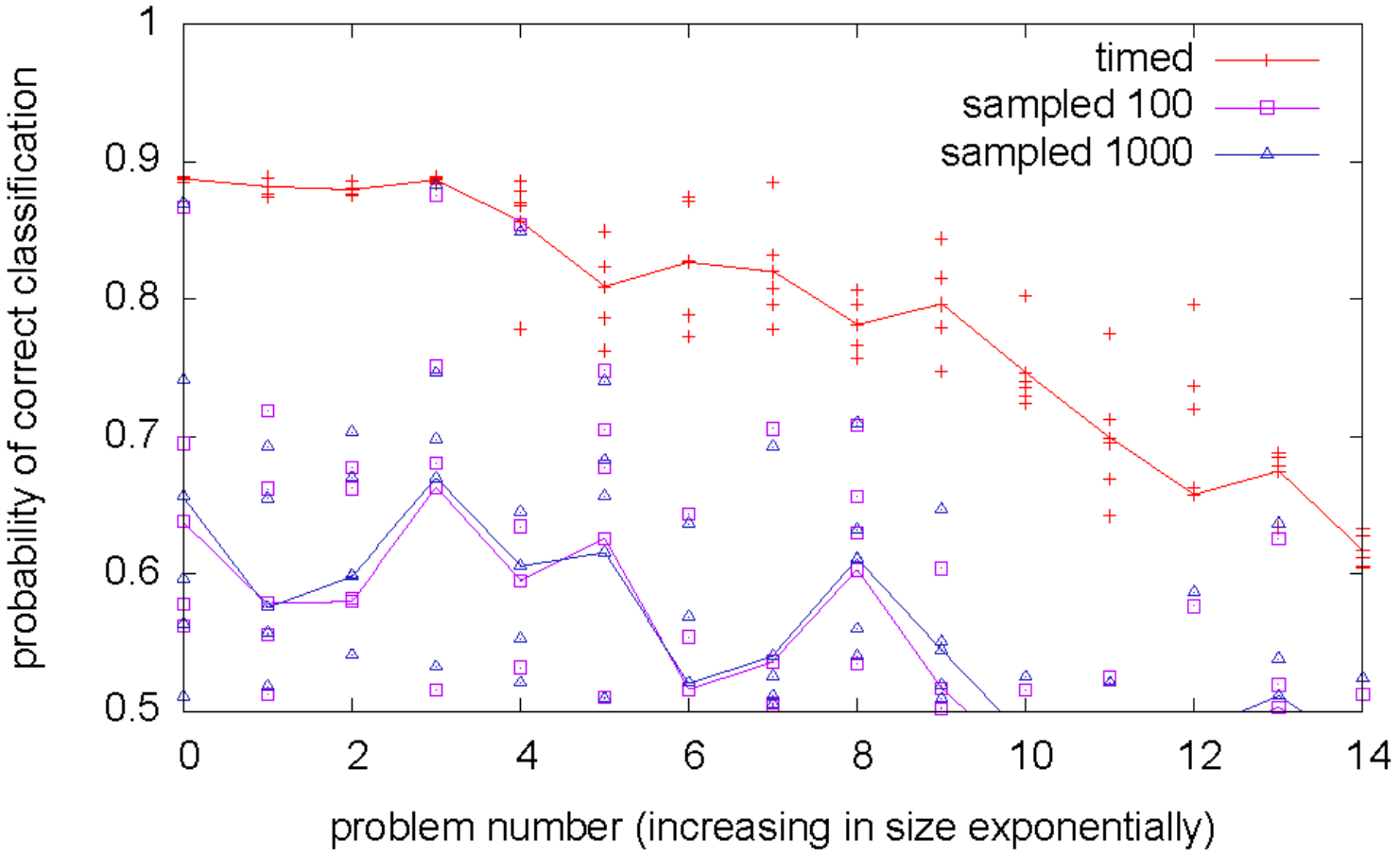


TU Delft

Results - 2000 samples



Results - 10000 samples



TU Delft

Conclusions