Pattern Discovery in Colored Strings

*Zsuzsanna Lipták[1] , Simon J. Puglisi[2] , and **Massimiliano Rossi**[3]*

[1] University of Verona, Department of Computer Science.
[2] University of Helsinki, Department of Computer Science.
[3] University of Florida, Department of Computer & Information Science & Engineering.

SEA 2020
16 Jun 2020, Catania (online)

Motivations – Assertion mining

Embedded Systems are everywhere

The design of embedded systems requires to evaluate the correctness of its functionalities. Usually done using **assertions** (Logic formulae).

Typically written by hand by the designers. It might take months to find a small and effective set of assertions.

Automatic extraction of **assertions** from simulation traces.

Motivations – Assertion mining

Simulation trace

T	i_1	i_2	i_3	o_1	o_2
1	0	1	0	0	0
2	1	1	0	1	0
3	0	1	0	0	0
4	1	1	0	1	1
5	0	1	0	0	0
6	1	1	0	1	0
7	1	0	1	1	1
8	0	1	0	1	0
9	1	1	0	0	0
10	0	1	0	0	0
11	1	0	1	1	1

Simulation trace

T	i_1	i_2	i_3	o_1	o_2
1	0	1	0	0	0
2	1	1	0	1	0
3	0	1	0	0	0
4	1	1	0	1	1
5	0	1	0	0	0
6	1	1	0	1	0
7	1	0	1	1	1
8	0	1	0	1	0
9	1	1	0	0	0
10	0	1	0	0	0
11	1	0	1	1	1

Motivations – Assertion mining

Simulation trace

T	i_1	i_2	i_3	o_1	o_2
1	0	1	0	0	0
2	1	1	0	1	0
3	0	1	0	0	0
4	1	1	0	1	1
5	0	1	0	0	0
6	1	1	0	1	0
7	1	0	1	1	1
8	0	1	0	1	0
9	1	1	0	0	0
10	0	1	0	0	0
11	1	0	1	1	1

Input alphabet

i_1	i_2	i_3	Σ
0	1	0	A
1	0	1	B
1	1	0	C

Output alphabet

o_1	o_2	Γ
0	0	X
1	0	Y
1	1	Z

Colored string

X	Y	X	Z	X	Y	Z	Y	X	X	Z
A	C	A	C	A	C	B	A	C	A	B
1	2	3	4	5	6	7	8	9	10	11

Colored Strings

Definition

Colored strings are strings where each character is assigned one of a finite set of colors.

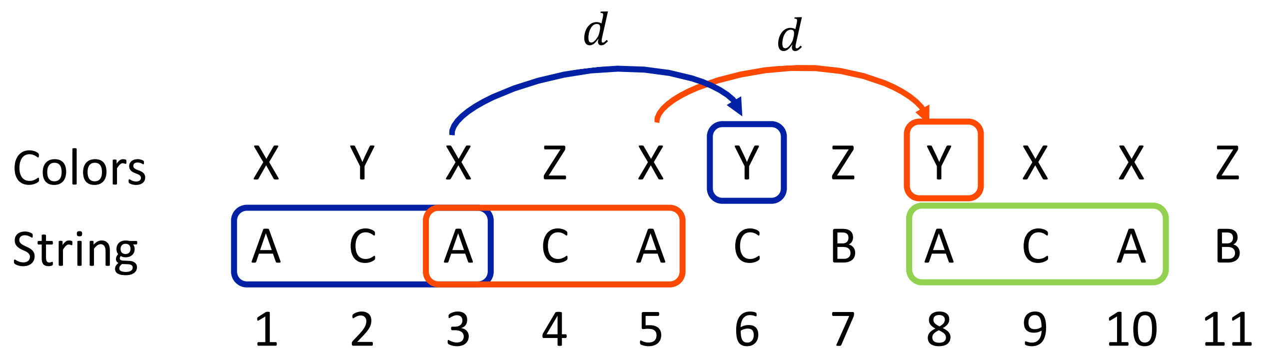
Objective

We want to find patterns in the string that always occur with the same color at a certain distance.

Colors	X	Y	X	Z	X	Y	Z	Y	X	X	Z
String	A	C	A	C	A	C	B	A	C	A	B
	1	2	3	4	5	6	7	8	9	10	11

We say that ACA is (Y,3)-unique.

Pattern Discovery



We say that ACA is (Y,3)-unique.

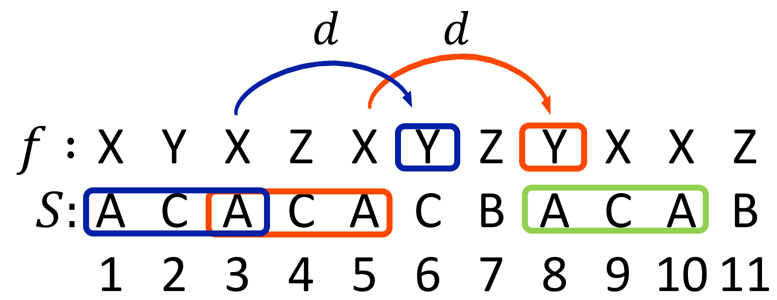
Problem

Given a colored string S and a color Y, report all pairs (T,d) such that T is (Y,d)-unique substring of S.

Note

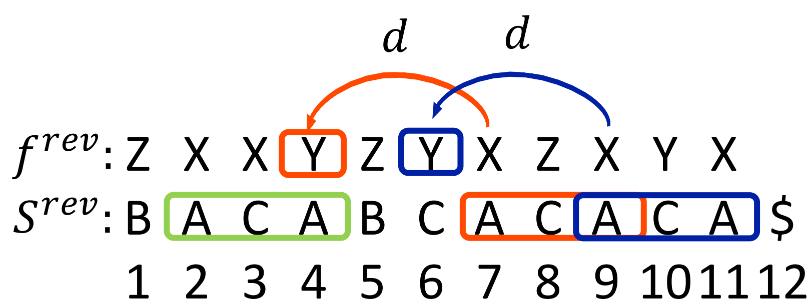
Although this problem is simpler than the assertion mining problem, the solution to our problem contains all the information, possibly filtered, to recover the desired set of minimal assertions in a second stage.

Discovery all (y, d)-unique substrings

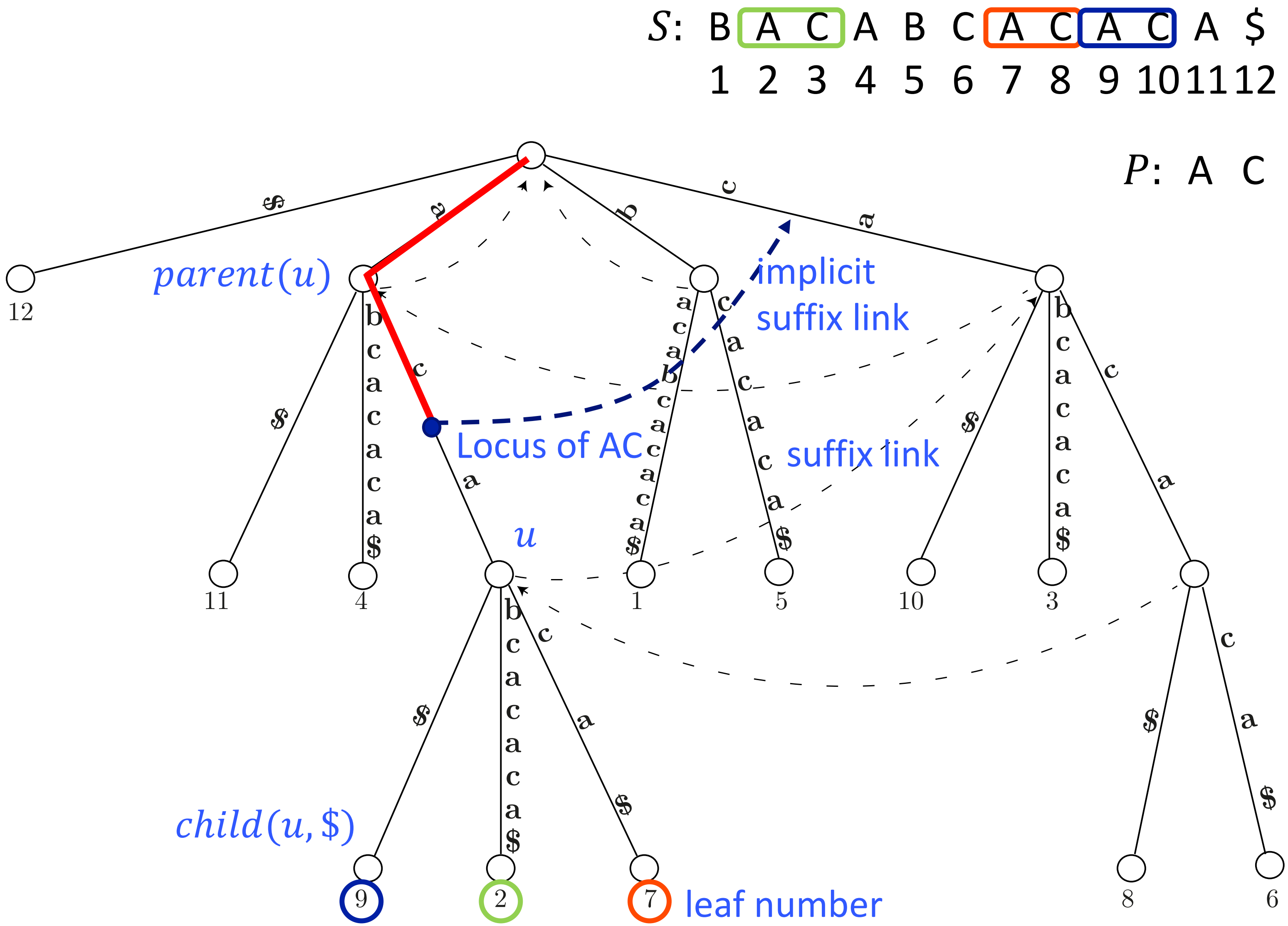


We need to check all occurrences of a substring of S. To keep the space contained, we use dedicated string data structures, i.e. **Suffix trees.**

Since the delay is measured from the end of the substring, it is convenient to think in terms of prefixes, i.e. **Suffixes of the reverse string.**



Suffix tree

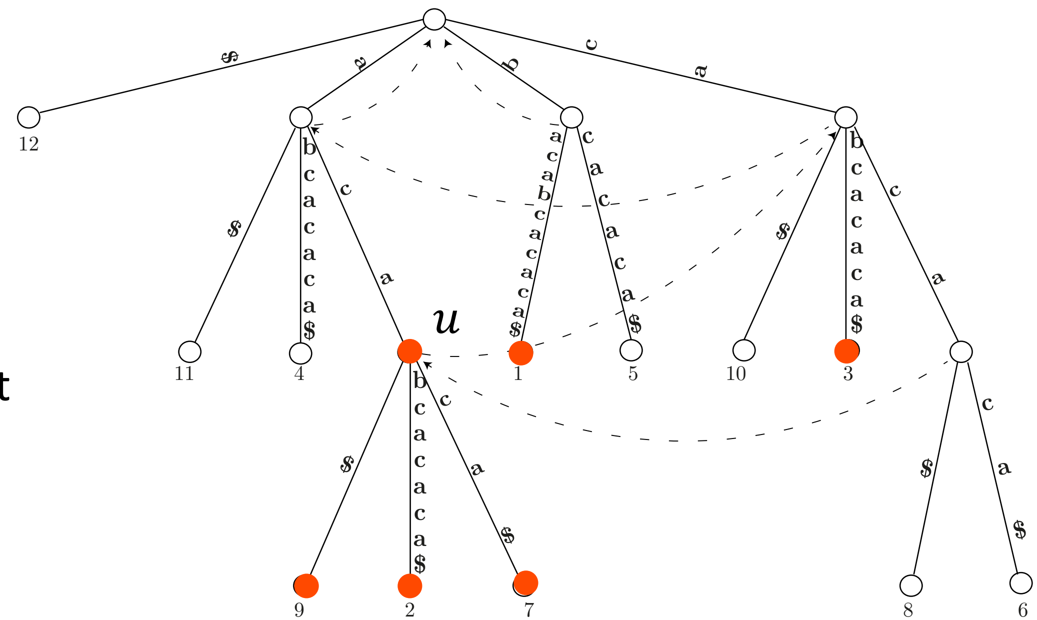

S: B A C A B C A C A C A $
1 2 3 4 5 6 7 8 9 10 11 12
P: A C
parent(u)
implicit suffix link
Locus of AC
suffix link
u
child(u, $)
leaf number
12
11
4
1
5
10
3
9
2
7
8
6

Discovery all (y, d)-unique substrings

f : X Y X Z X Y Z Y X X Z
S: A C A C A C B A C A B
1 2 3 4 5 6 7 8 9 10 11

f^{rev}: Z X X Y Z Y X Z X Y X
S^{rev}: B A C A B C A C A C A $
1 2 3 4 5 6 7 8 9 10 11 12

1. Build the **suffix tree** of S^{rev}
2. Color a leaf if:
   - Either $ln \leq d$
   - or $f(ln - d) = y$
3. Color an internal node if:
   - All children are colored.
4. If a node u is colored, output all strings represented along the incoming edge of u.



Output: …, CA, ACA, …

Runs in $O(n^3)$ time.

$d = 3$
$y =$ Y

Minimal (y, d)-unique substrings

d d

X	Y	X	Z	X	$\boxed{\text{Y}}$	Z	$\boxed{\text{Y}}$	X	X	Z
A	$\boxed{\text{C A}}$		$\boxed{\text{C A}}$		C	B	A	$\boxed{\text{C A}}$		B
1	2	3	4	5	6	7	8	9	10	11

We say that CA is **minimal** (Y,3)-unique,
because A is **not** (Y,3)-unique and C is **not** (Y,4)-unique.

left-minimality right-minimality

Problem

Given a colored string S and a color Y, report all pairs (T,d) such that T is **minimal** (Y,d)-unique substring of S.

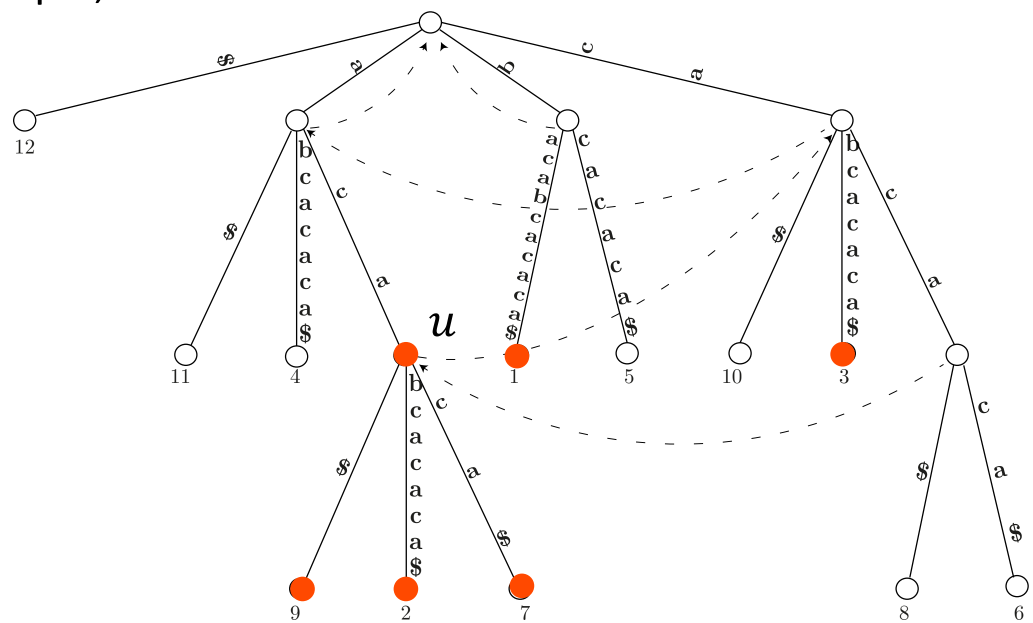
Discovery all minimal (y, d)-unique substrings

f:	X	Y	X	Z	X	Y	Z	Y	X	X	Z
S:	A	C	A	C	A	C	B	A	C	A	B
	1	2	3	4	5	6	7	8	9	10	11

f^{rev}:	Z	X	X	Y	Z	Y	X	Z	X	Y	X	
S^{rev}:	B	A	C	A	B	C	A	C	A	C	A	\$
	1	2	3	4	5	6	7	8	9	10	11	12

We say that CA is **minimal** (Y,3)-unique, because

1. A is **not** (Y,3)-unique **and**
2. C is **not** (Y,4)-unique.



(left minimality)

Parent of AC is not colored.

Suffix link of AC is not colored for $d = 4$. (right minimality)

Process d from 11 downto 0

Output: …, CA, …

Runs in $O(n^2)$ time.

$d = 3$

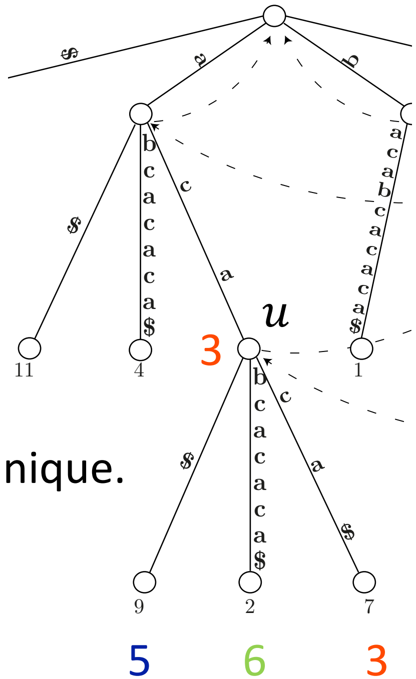
$y = \mathrm{Y}$

Skipping Algorithm

Skipping algorithm

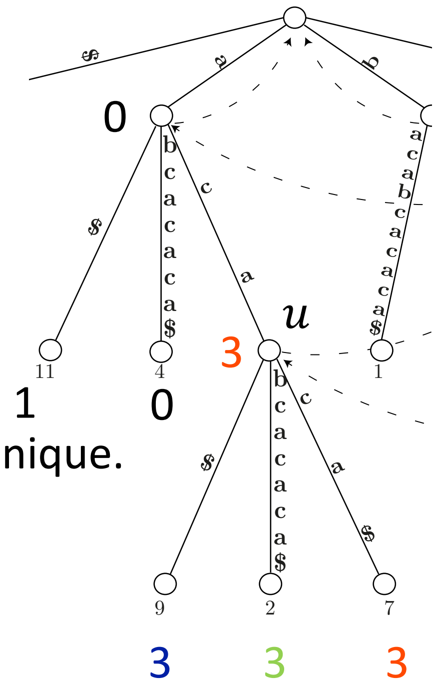
Given a node u and an integer ℓ, $h(u,\ell)$ is the largest delay $d < \ell$ such that the corresponding string can be (y,d)-unique.

$\ell = 7$
$y = \mathrm{Y}$



u can be $(\mathrm{Y},3)$-unique.

$\ell = 4$
$y = \mathrm{Y}$



u is $(\mathrm{Y},3)$-unique.

f^{rev}: Z X X Y Z Y X Z X Y X
S^{rev}: B A C A B C A C A C A $
1 2 3 4 5 6 7 8 9 10 11 12

f^{rev}: Z X X Y Z Y X Z X Y X
S^{rev}: B A C A B C A C A C A $
1 2 3 4 5 6 7 8 9 10 11 12

Skipping algorithm

- We discover the strings for d from n downto 0. (right minimality)
- For each node u, we keep the value $h(u, d+1)$ updated.
- We find a node u such that:
  1. u has the largest value $h(u, d+1)$;
  2. u has priority on its children. (left minimality)
- We check if u is right minimal, and if so, we report it.
- We update:
  1. the value of all nodes v in the subtree rooted on u to $h(v, d)$
  2. the value of all ancestors v of u to $h(v, d)$

Maximum-oriented indexed priority queue

Runs in $O(n^2 \log n)$ time.

Output restrictions

Output restrictions

We restrict the output to (y, d)-unique substrings with at least two occurrences followed by y.

f : X Y X Z X Y Z Y X X Z
S: A C A C A C B A C A B
1 2 3 4 5 6 7 8 9 10 11
$d = 3$

Considered

f : X Y X Z X Y Z Y X X Z
S: A C A C A C B A C A B
1 2 3 4 5 6 7 8 9 10 11
$d = 7$

Not considered

Including this consideration as part of the problem, we can modify the computation of $h(u, d)$, when all children of u are leaves.

Experimental results

Experimental results

Algorithms:

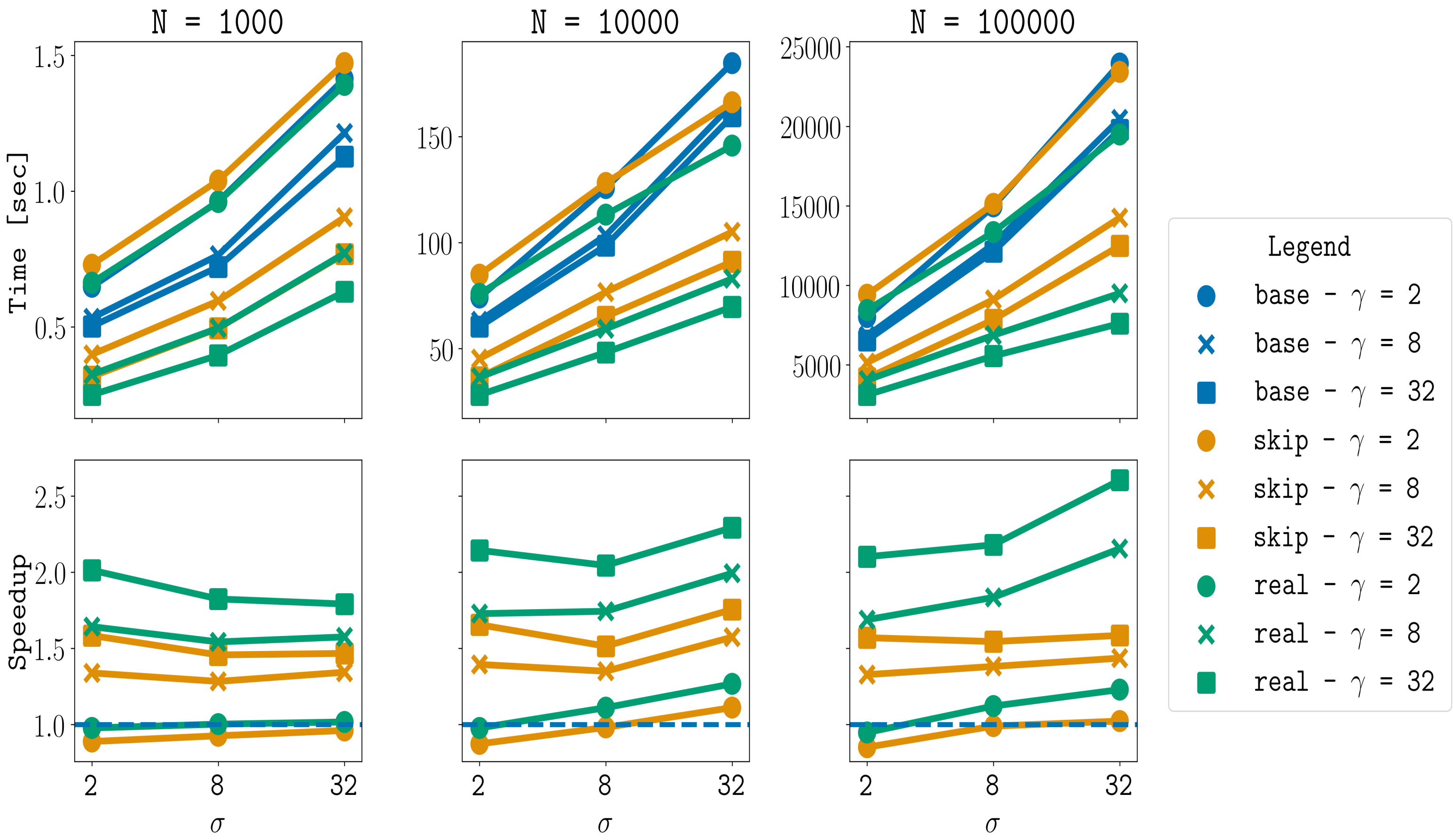
- **Baseline**: Suffix tree based algorithm.
- **Skipping**: Skipping algorithm.
- **Real:** Skipping with output restrictions as part of the problem.

Data:

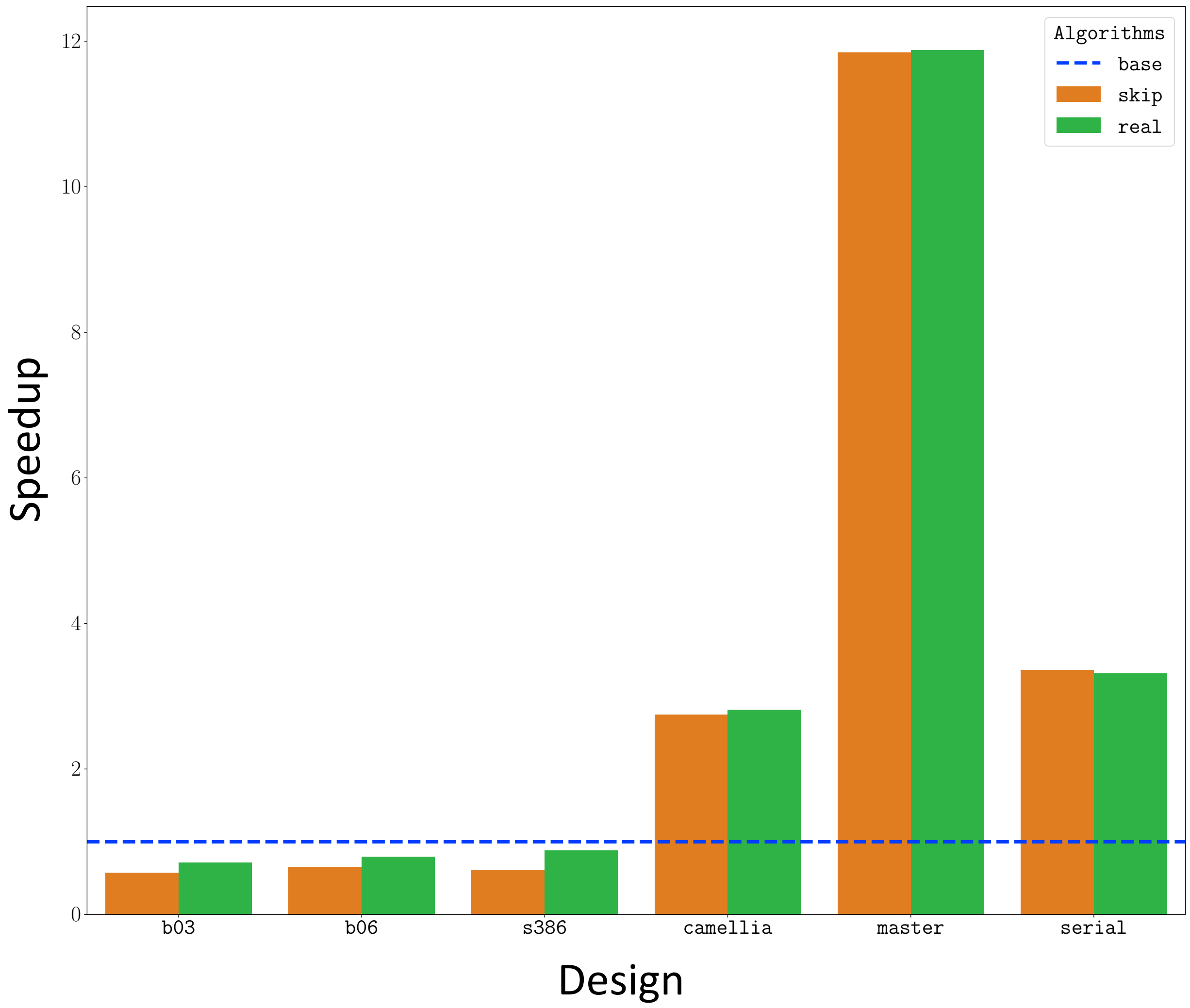
1. **Synthetic data:** Randomly generated data varied:
   - string length $n = 100, \mathbf{1000}, \mathbf{10000}, \mathbf{100000}$,
   - alphabet size $= \mathbf{2}, 4, \mathbf{8}, 16, \mathbf{32}$,
   - number of colors $= \mathbf{2}, 4, \mathbf{8}, 16, \mathbf{32}$.
2. **Real data:** Simulation on a set of established benchmarks in embedded systems verification.

Design	Description	PIs	POs	n	σ	γ	n_γ
b03	Resource arbiter	6	4	100000	17	5	3210
b06	Interrupt handler	4	6	100000	5	4	44259
s386	Synthetized controller	9	7	100000	129	2	8290
camelia	Symmetric key block cypher	262	131	103615	70	224	2292
serial	Serial data transmitter	11	2	100000	118	2	16353
master	Wishbone bus master	134	135	100000	417	80	759

Synthetic data



Real data


Speedup
Design
Algorithms
base
skip
real
0
2
4
6
8
10
12
b03
b06
s386
camellia
master
serial

Thank you for your attention!