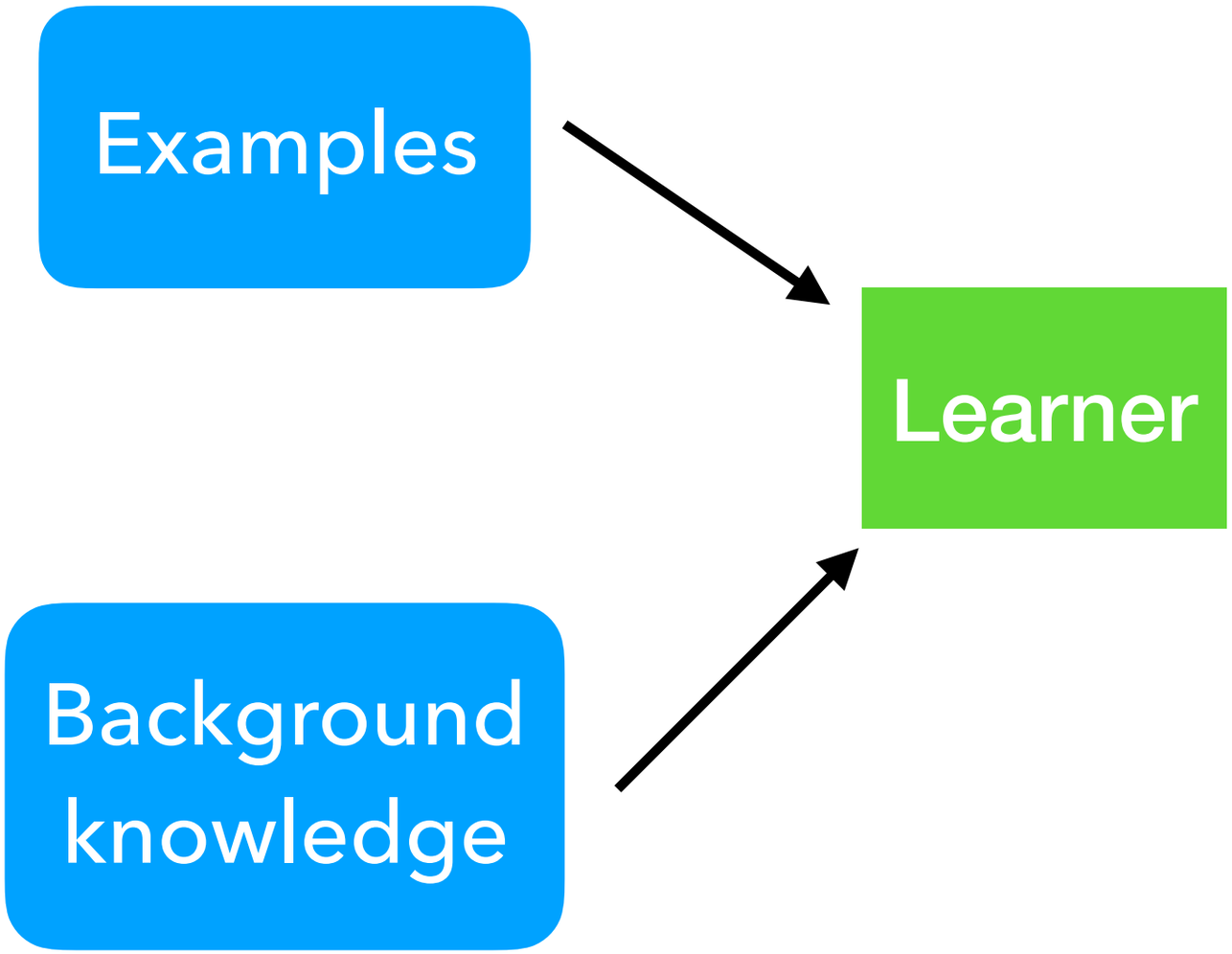
Forgetting to learn logic programs

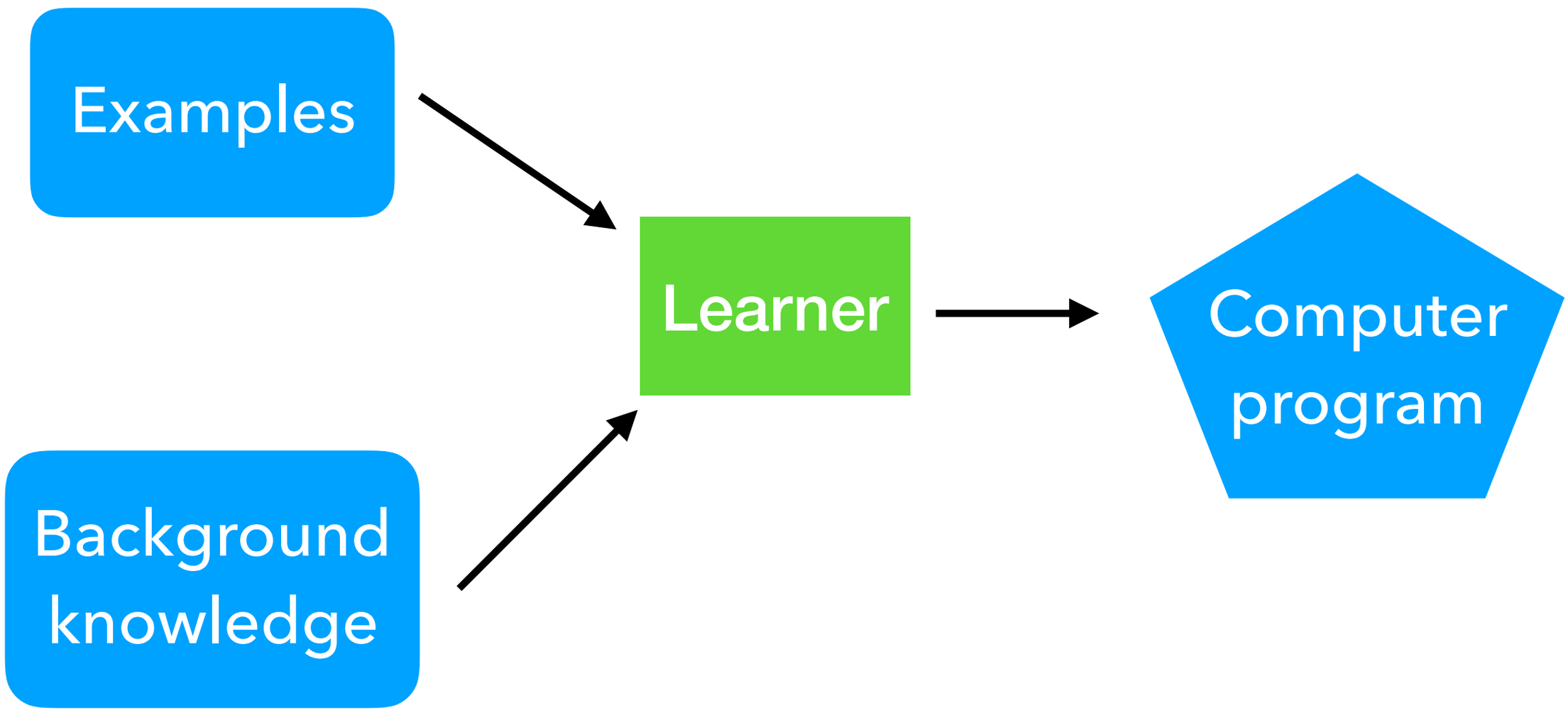
Andrew Cropper

University of Oxford

Program induction/synthesis



Program induction/synthesis



Examples

input	output
dog	g
sheep	p
chicken	**?**

Examples

input	output
dog	g
sheep	p
chicken	**?**

Background knowledge

head

tail

empty

Examples

input	output
dog	g
sheep	p
chicken	**?**

Background knowledge

head

tail

empty

```
def f(a):
    t = tail(a)
    if empty(t):
        return head(a)
    return f(t)
```

Examples

input	output
dog	g
sheep	p
chicken	**n**

Background knowledge

head

tail

empty

```
def f(a):
    t = tail(a)
    if empty(t):
        return head(a)
    return f(t)
```

Examples

input	output
dog	g
sheep	p
chicken	**n**

Background knowledge

head

tail

empty

```
f(A,B):-tail(A,C),empty(C),head(A,B).
f(A,B):-tail(A,C),f(C,B).
```

Background knowledge defines the hypothesis space

Where does background knowledge come from?

Hand-crafted rules

[almost every approach]

Unsupervised learning

ALPS [Dumančić et al. IJCAI 2019]

Playgol [Cropper. IJCAI 2019]

Supervised multi-task learning

Use knowledge gained from solving one problem to help solve a different problem

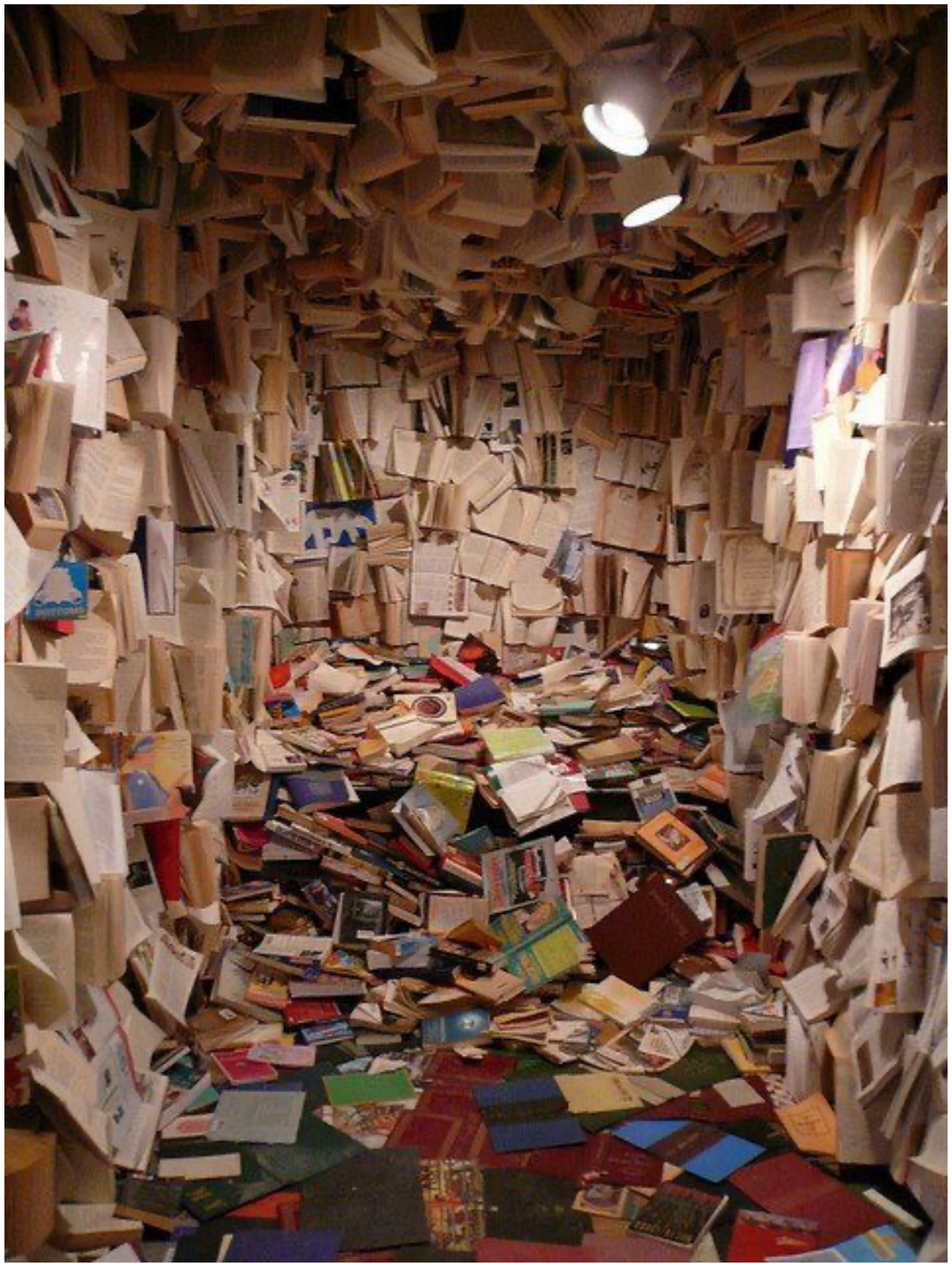
Metabias [Lin et al. ECAI 2014]
Dreamcoder [Ellis et al. NIPS 2018]

Why does it work?

We increase branching but reduce depth

Size Bound	Dependent Learning	Independent Learning
Time Out	9	17, 4, 9, 5
5	3	3, 13, 11
4	5, 7, 8, 4, 6, 12, 13, 11	1, 6, 7, 8, 12
3	1, 10, 17	10, 15
2	2, 15	2
1	14, 16	14, 16

Problem: big branching factor



Idea

Forget things

ILP problem

Given:

- background knowledge **B**
- positive examples **E+**
- negative examples **E-**

ILP problem

Return:

A hypothesis **H** that with **B** entails **E+** and not **E+**

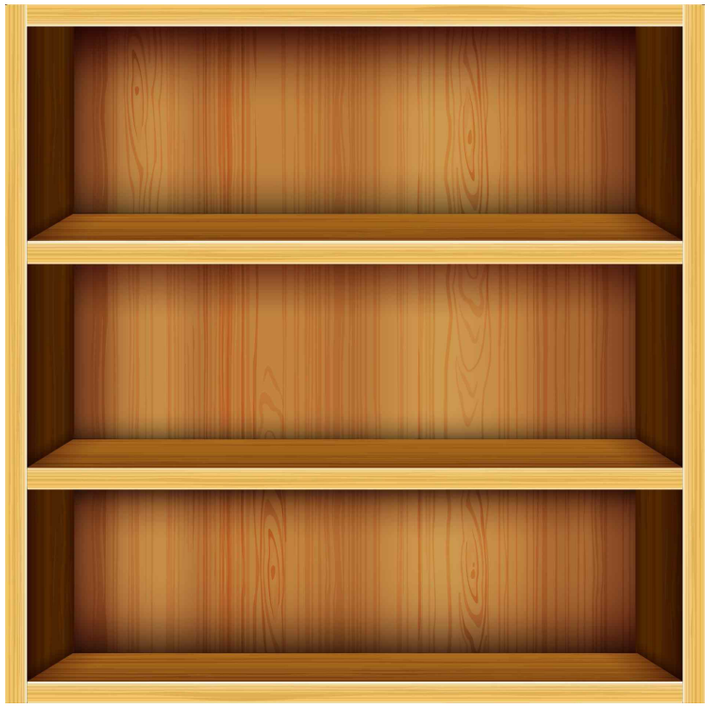
Forgetting problem

Given background knowledge **B**

Return **B’** ⊂ **B** from which you can still learn the target hypothesis

Why?

Reduce branching and sample complexity



How?

Forgetgol, a multi-task ILP system based on Metagol, which takes a forgetting function as input

Forgetgol

Continually **expands** and **shrinks** its hypothesis space

Syntactical forgetting
(lossless)

1. Unfold each induced clause to remove invented predicate symbols
2. Check whether a syntactically duplicate clause already exists

Statistical forgetting

Assigns a cost to each clause based on:

1. How difficult it was to learn
2. How likely it is to be reused

Does it work?

Q. Can forgetting improve learning performance?

We compare:

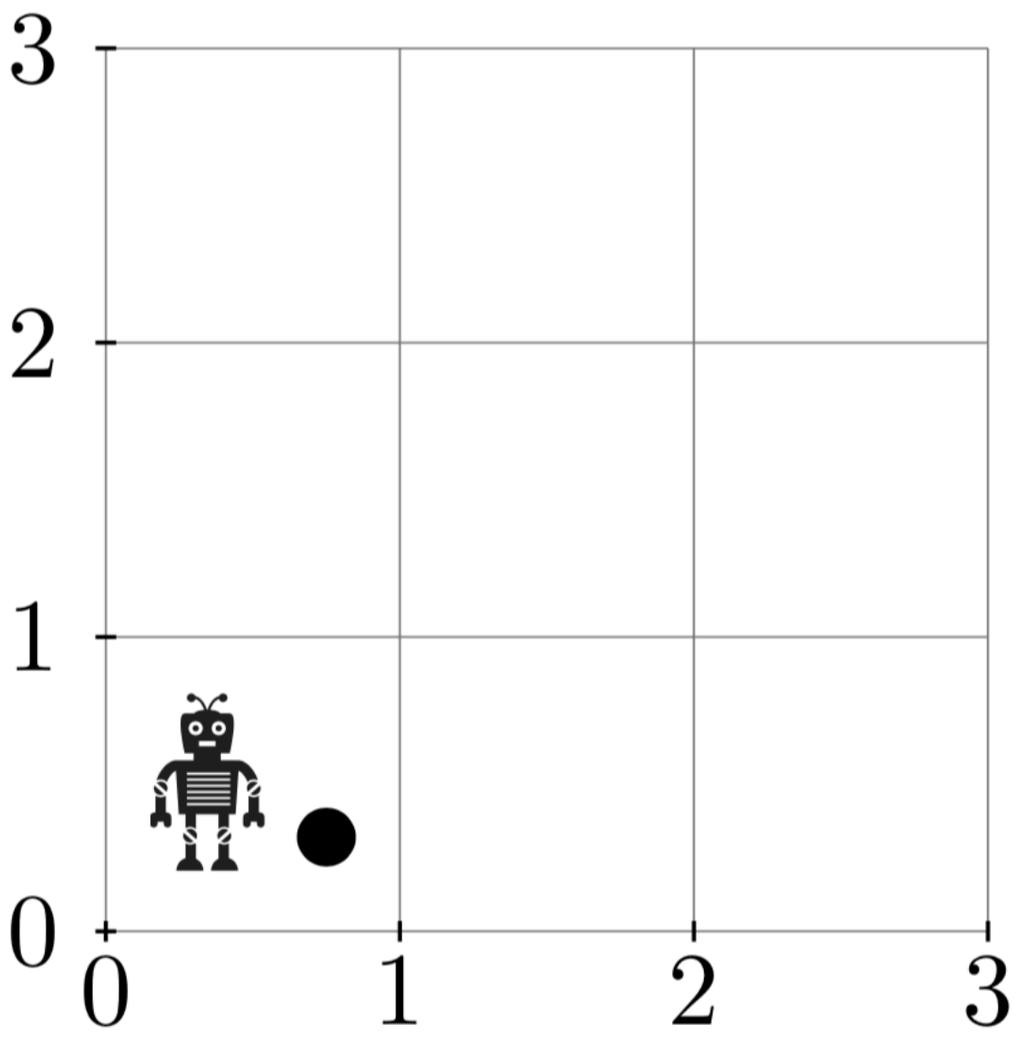
Metabias: remember everything

Metagol: remember nothing

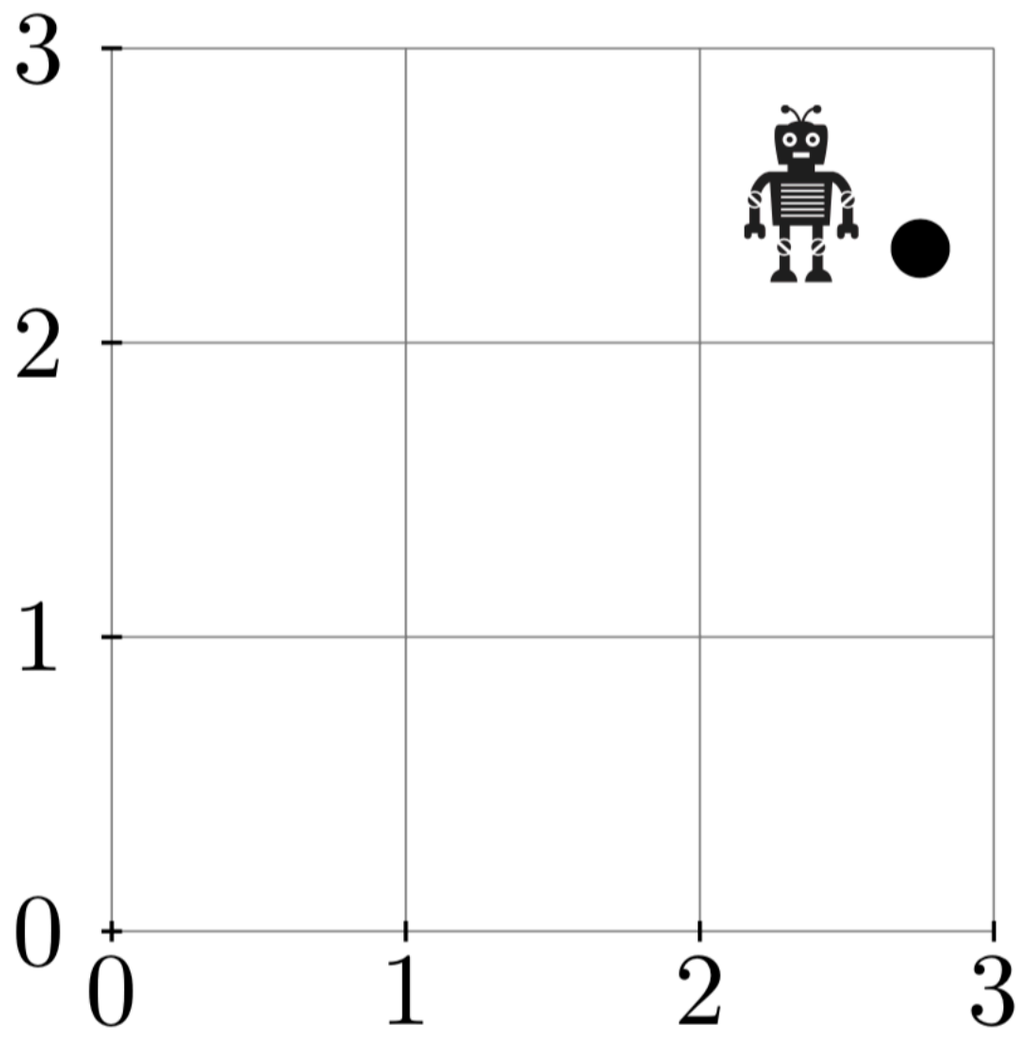
Forgetgol$_{syn}$: syntactical forgetting

Forgetgol$_{stat}$: statistical forgetting

Robot planning



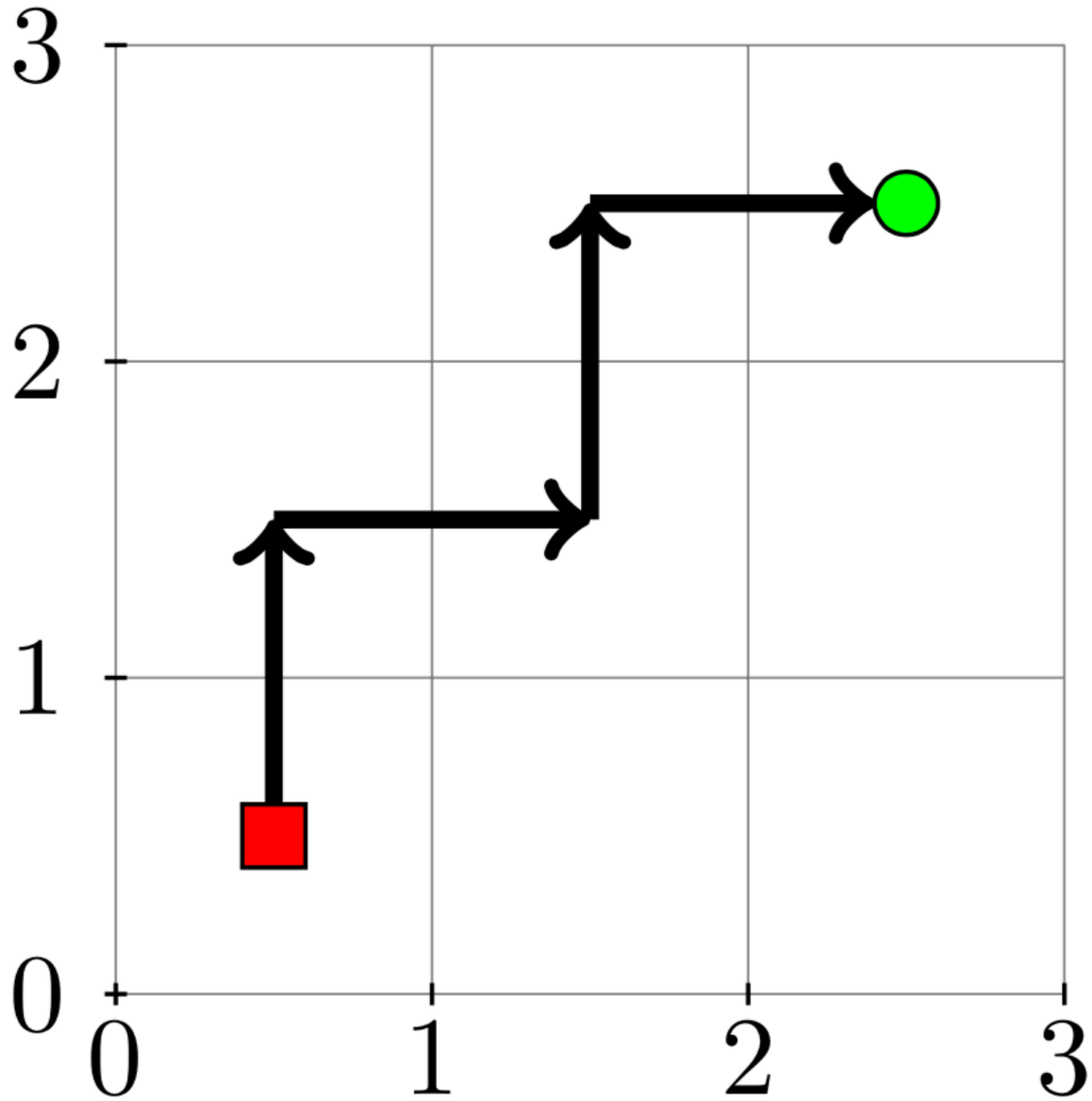
(a) Initial state



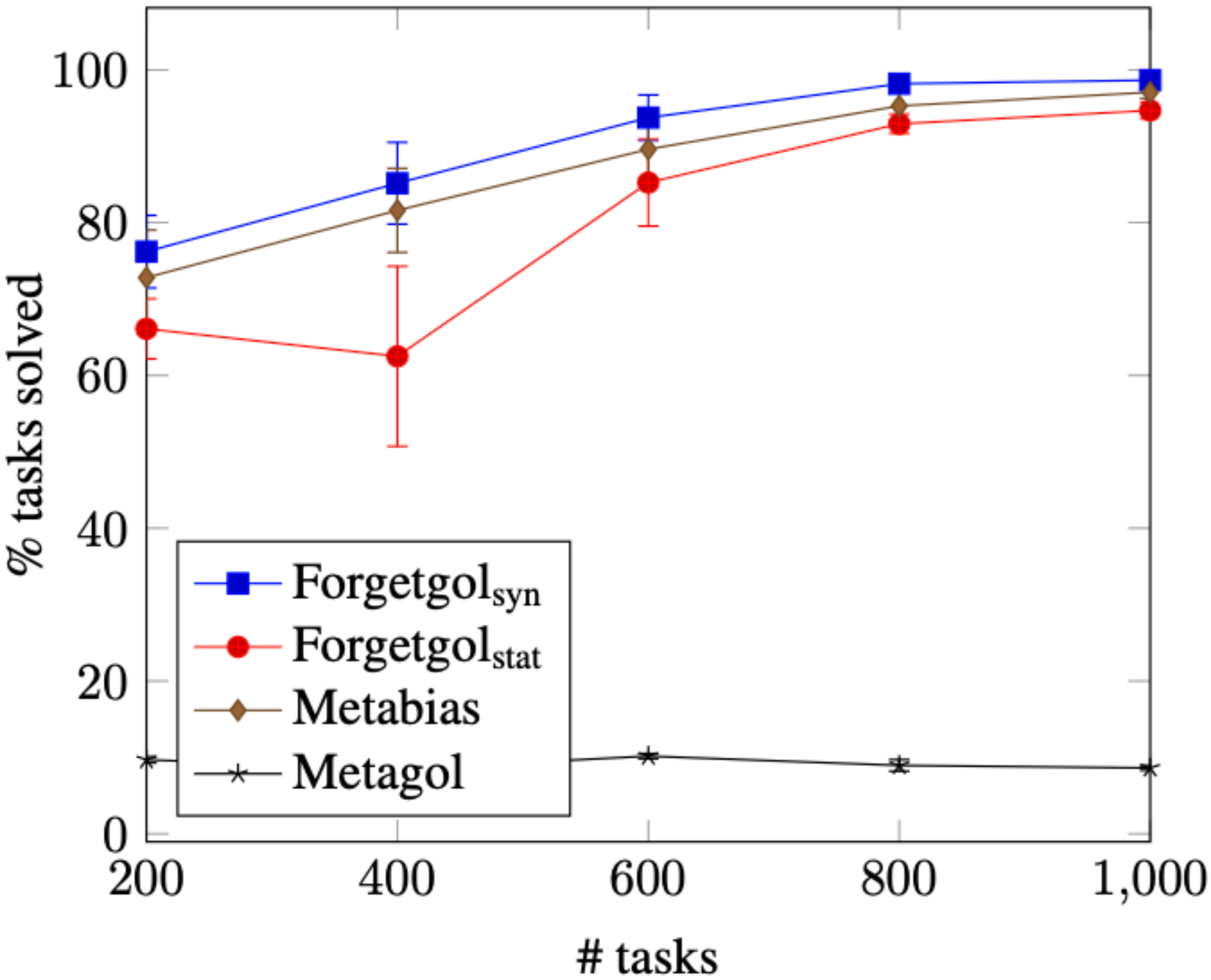
(b) Final state

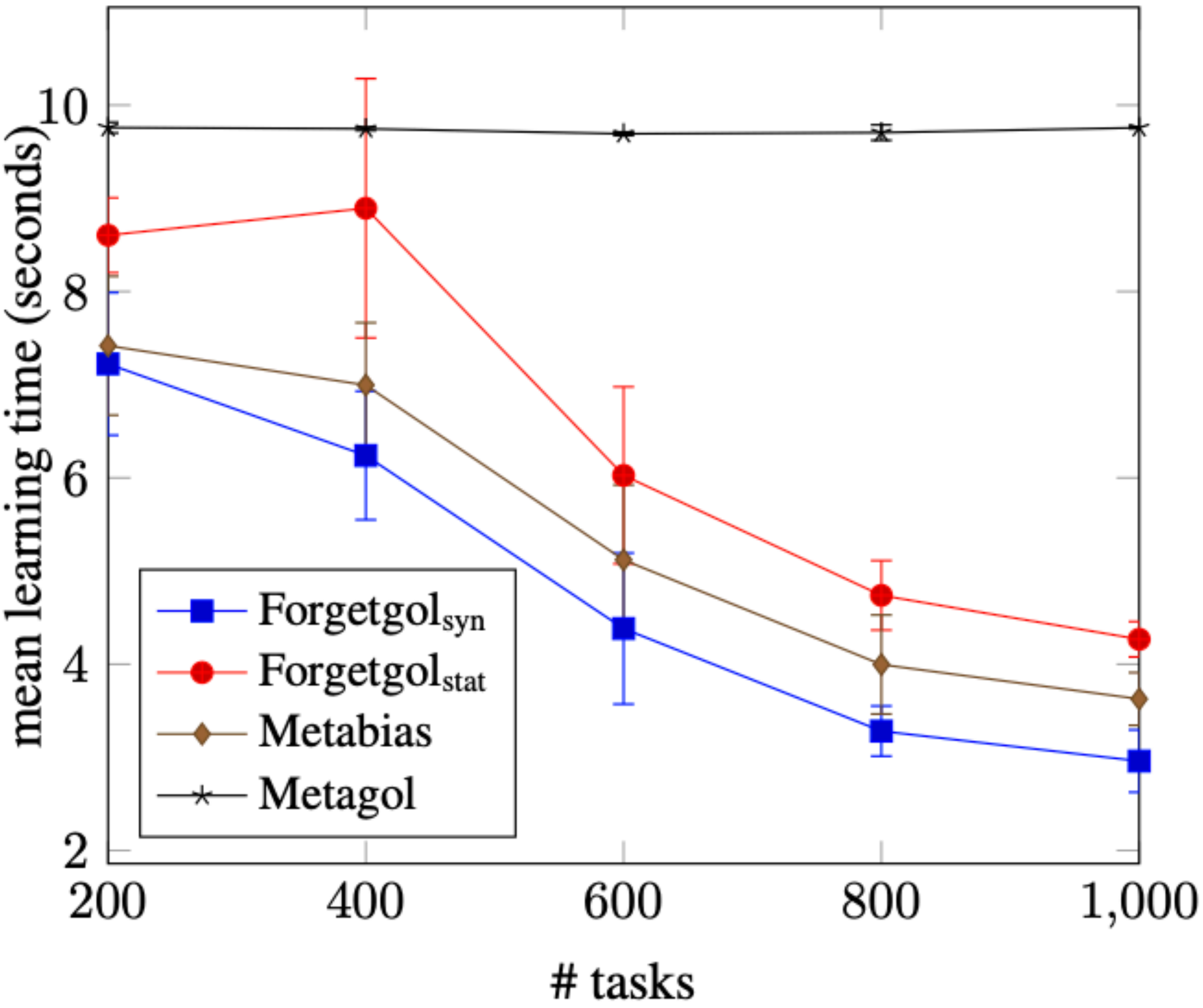
```
f(A,B):-
      grab(A,C),
      f1(C,D),
      f1(D,E),
      drop(E,B).
f1(A,B):-
      up(A,C),
      right(C,B).
```

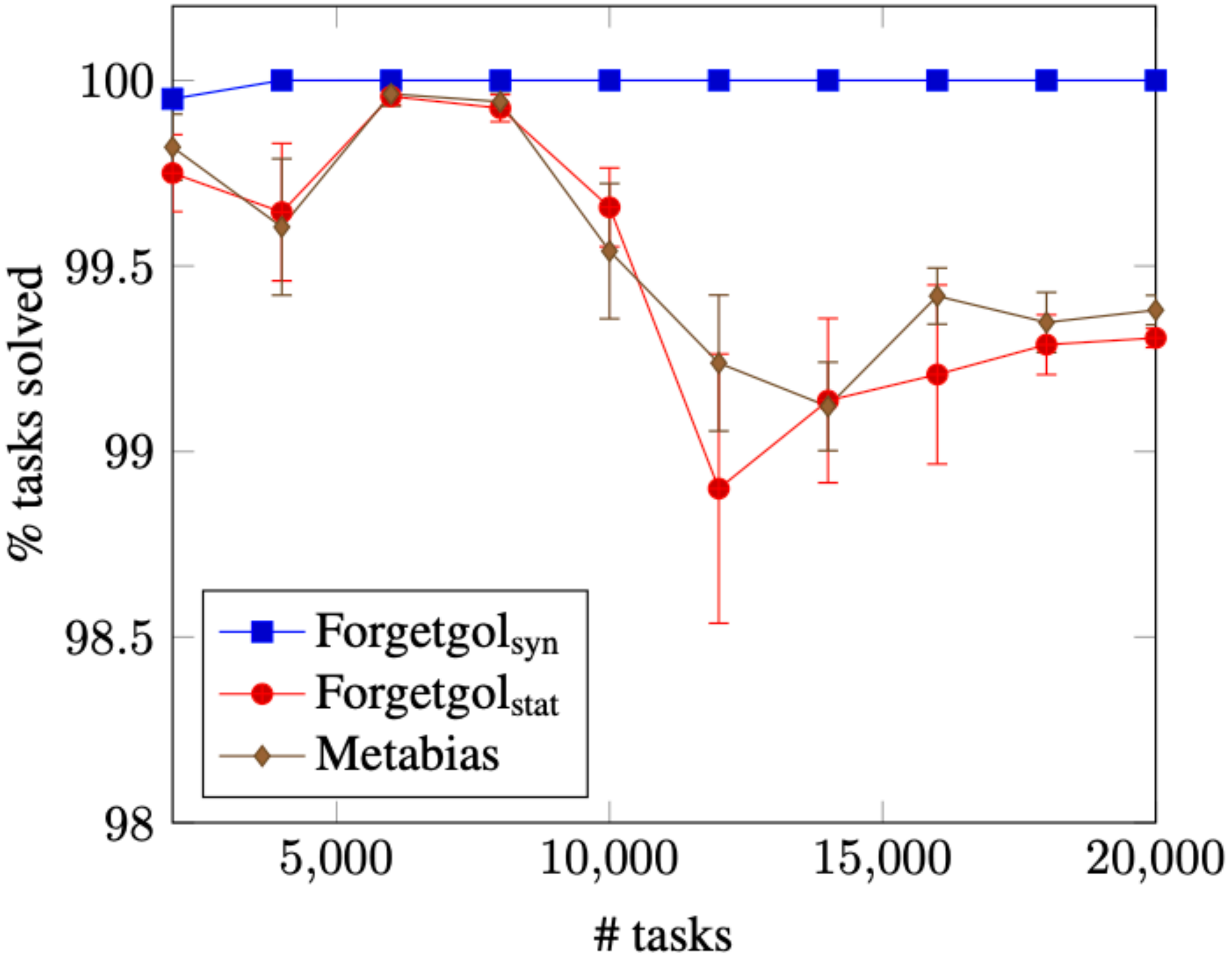
(a) Program

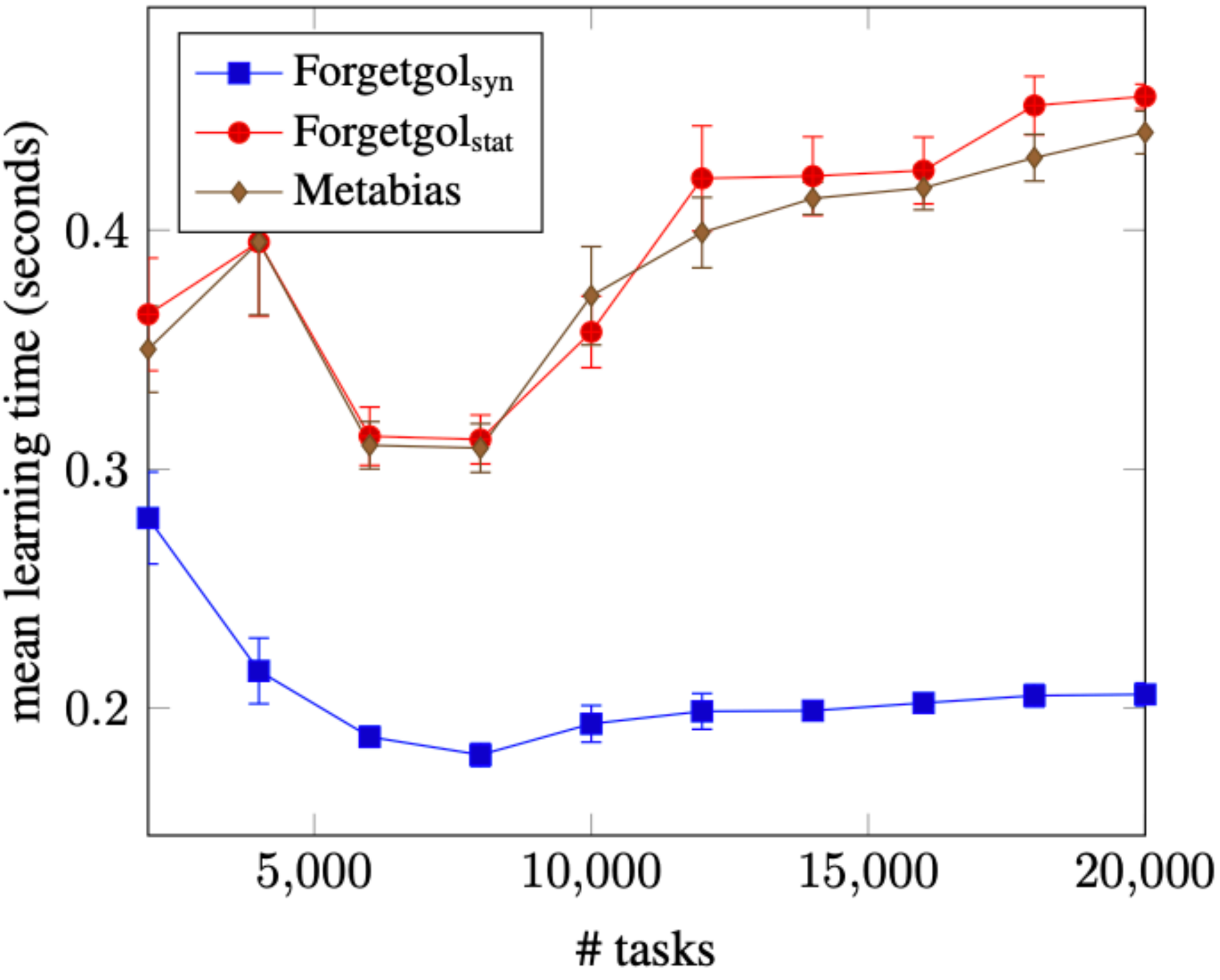


(b) Plan


100
80
60
40
20
0
% tasks solved
200
400
600
800
1,000
tasks
Forgetgol$_{syn}$
Forgetgol$_{stat}$
Metabias
Metagol


mean learning time (seconds)
10
8
6
4
2
200
400
600
800
1,000
tasks
Forgetgol$_{syn}$
Forgetgol$_{stat}$
Metabias
Metagol

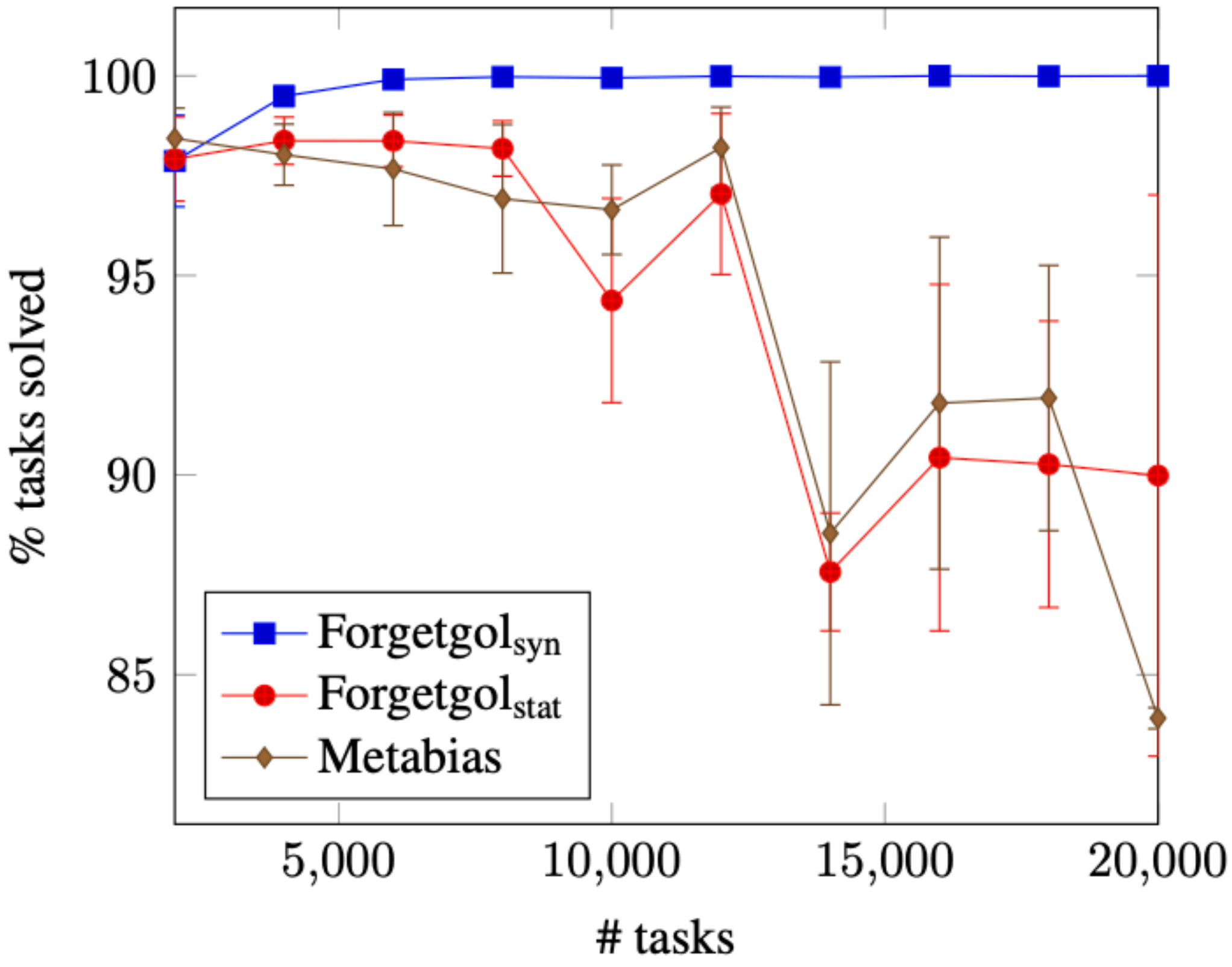

100
99.5
99
98.5
98
% tasks solved
5,000
10,000
15,000
20,000
tasks
Forgetgol$_{syn}$
Forgetgol$_{stat}$
Metabias

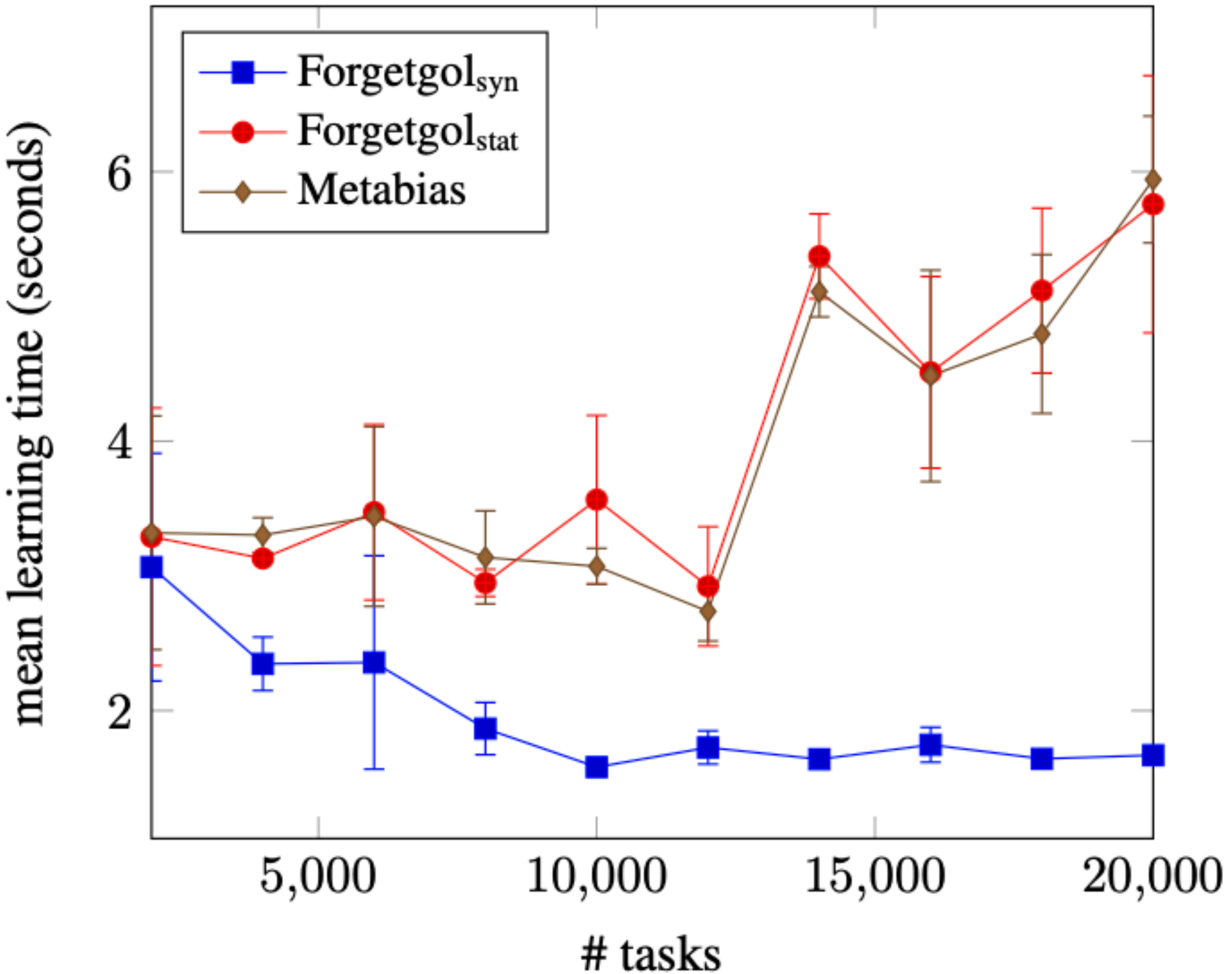

Forgetgol$_{syn}$
Forgetgol$_{stat}$
Metabias
mean learning time (seconds)
0.4
0.3
0.2
5,000
10,000
15,000
20,000
tasks

What happened?

Metabias rarely induces a program with more than two clauses because of program reuse

Lego building


100
95
90
85
% tasks solved
5,000
10,000
15,000
20,000
tasks
Forgetgol$_{syn}$
Forgetgol$_{stat}$
Metabias


Forgetgol$_{syn}$
Forgetgol$_{stat}$
Metabias
mean learning time (seconds)
6
4
2
5,000
10,000
15,000
20,000
tasks

What happened?

Less reuse (and greater search depth) so forgetting has more effect

Conclusions

Forgetting can improve learning performance when given >10,000 tasks, but, surprisingly, not by much

Limitations and future work

Better forgetting methods

Larger and more diverse datasets

Concept shift

Recency

Other program induction systems