A Closer Look at Pseudo-Polynomial Time and its Use in Real-Time Scheduling Theory

Sanjoy Baruah
Washington University in Saint Louis

Pontus Ekberg
Uppsala University

Dedicated to Wang Yi to Celebrate his Scientific Career

Come again?

Pseudo-polynomial time

An algorithm is pseudo-polynomial if its running time is polynomial in the *size n* of the input instance (in bits) and in the instance's *largest numerical parameter N*.

Come again?

Pseudo-polynomial time

An algorithm is pseudo-polynomial if its running time is polynomial in the *size n* of the input instance (in bits) and in the instance's *largest numerical parameter N*.

In scheduling, N would typically denote some measure of time and would not grow without bound.

Come again?

Pseudo-polynomial time

An algorithm is pseudo-polynomial if its running time is polynomial in the *size n* of the input instance (in bits) and in the instance's *largest numerical parameter N*.

In scheduling, N would typically denote some measure of time and would not grow without bound.

Say, $N = 2^{100}$ is not meaningful at human timescales.

Come again?

Pseudo-polynomial time

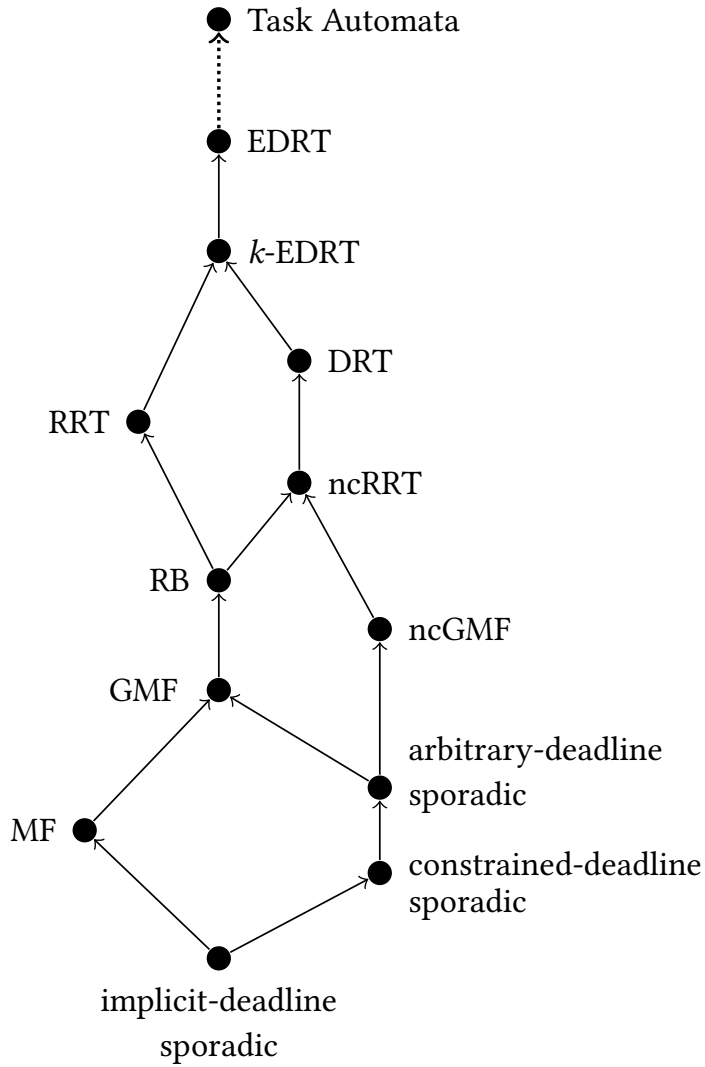
An algorithm is pseudo-polynomial if its running time is polynomial in the *size n* of the input instance (in bits) and in the instance's *largest numerical parameter N*.

In scheduling, N would typically denote some measure of time and would not grow without bound.

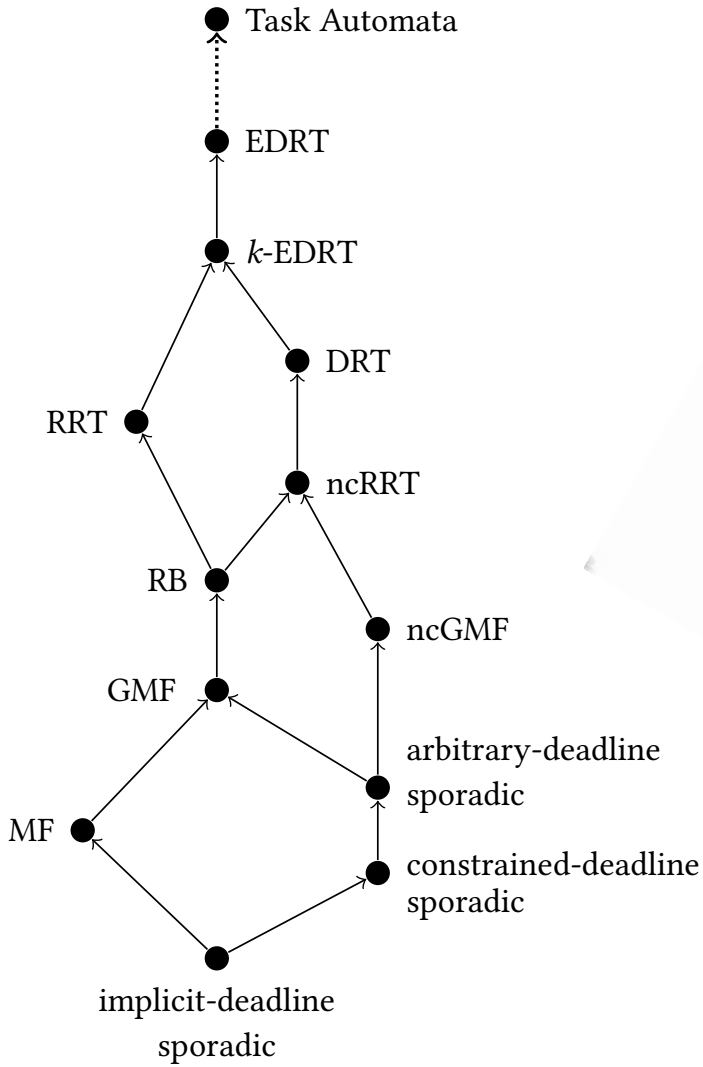
Say, $N = 2^{100}$ is not meaningful at human timescales.

Pseudo-polynomial time has traditionally been considered tractable in real-time scheduling.

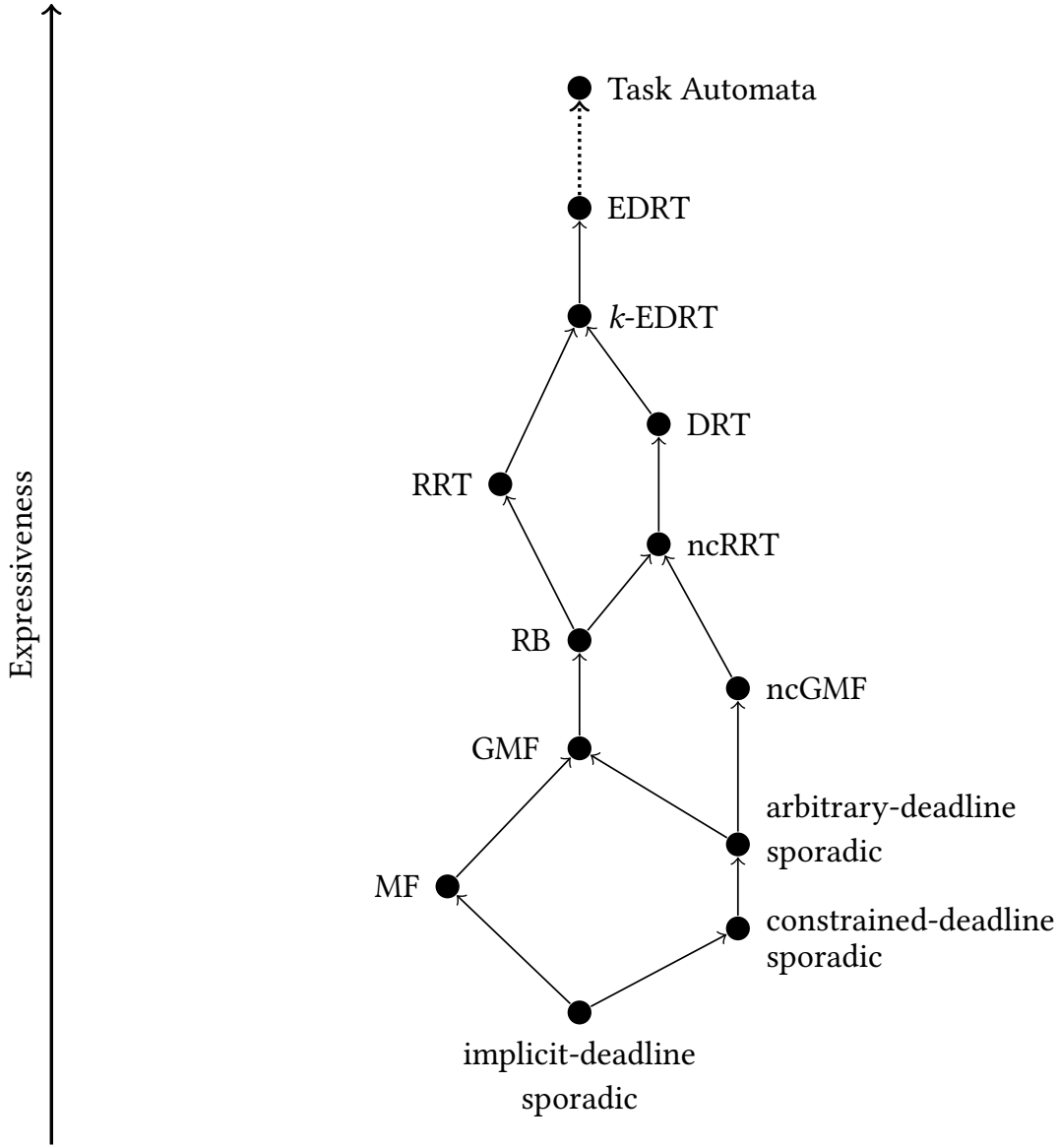
Pushing the boundary of pseudo-polynomial time



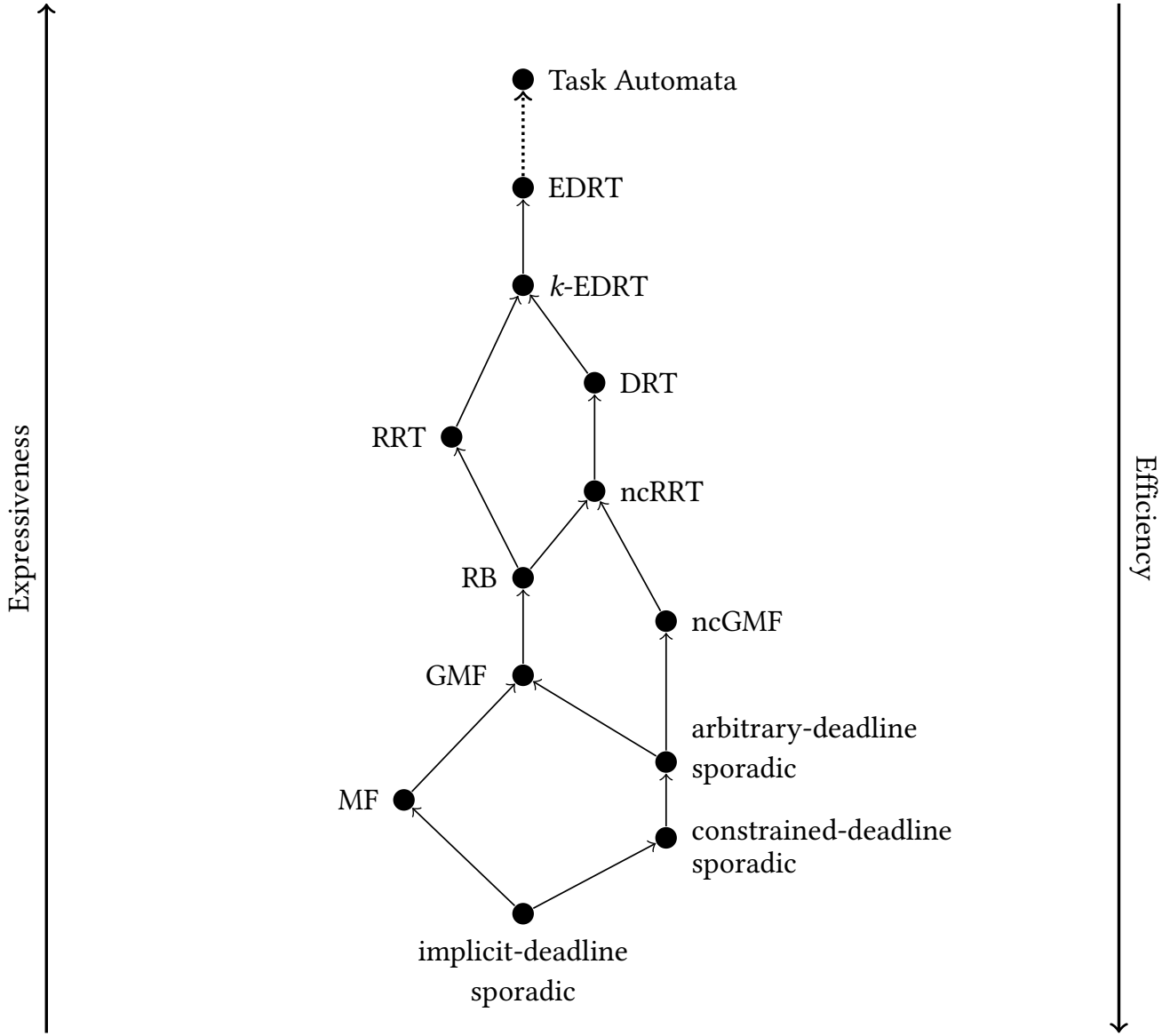
Pushing the boundary of pseudo-polynomial time



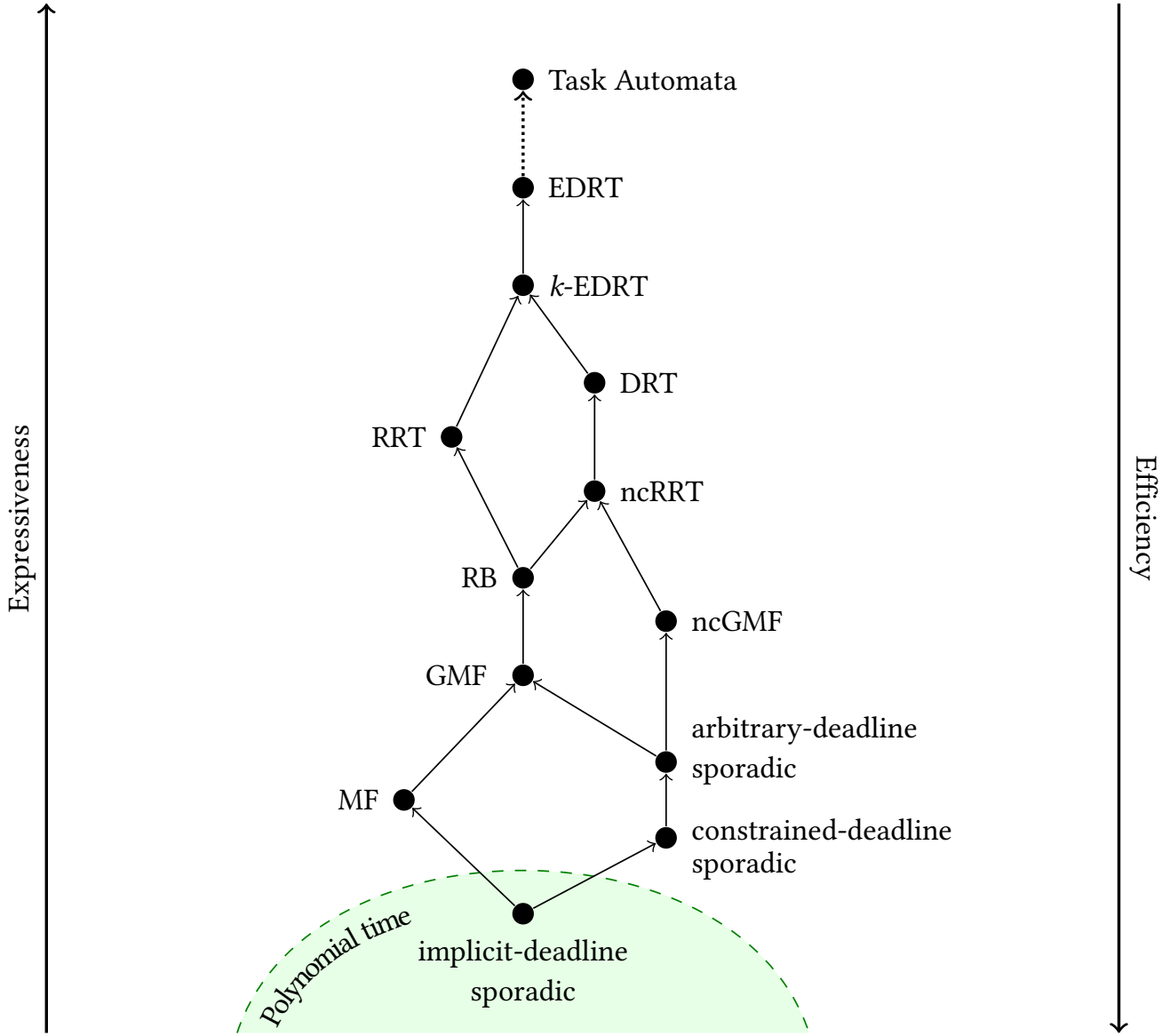
Pushing the boundary of pseudo-polynomial time



Pushing the boundary of pseudo-polynomial time

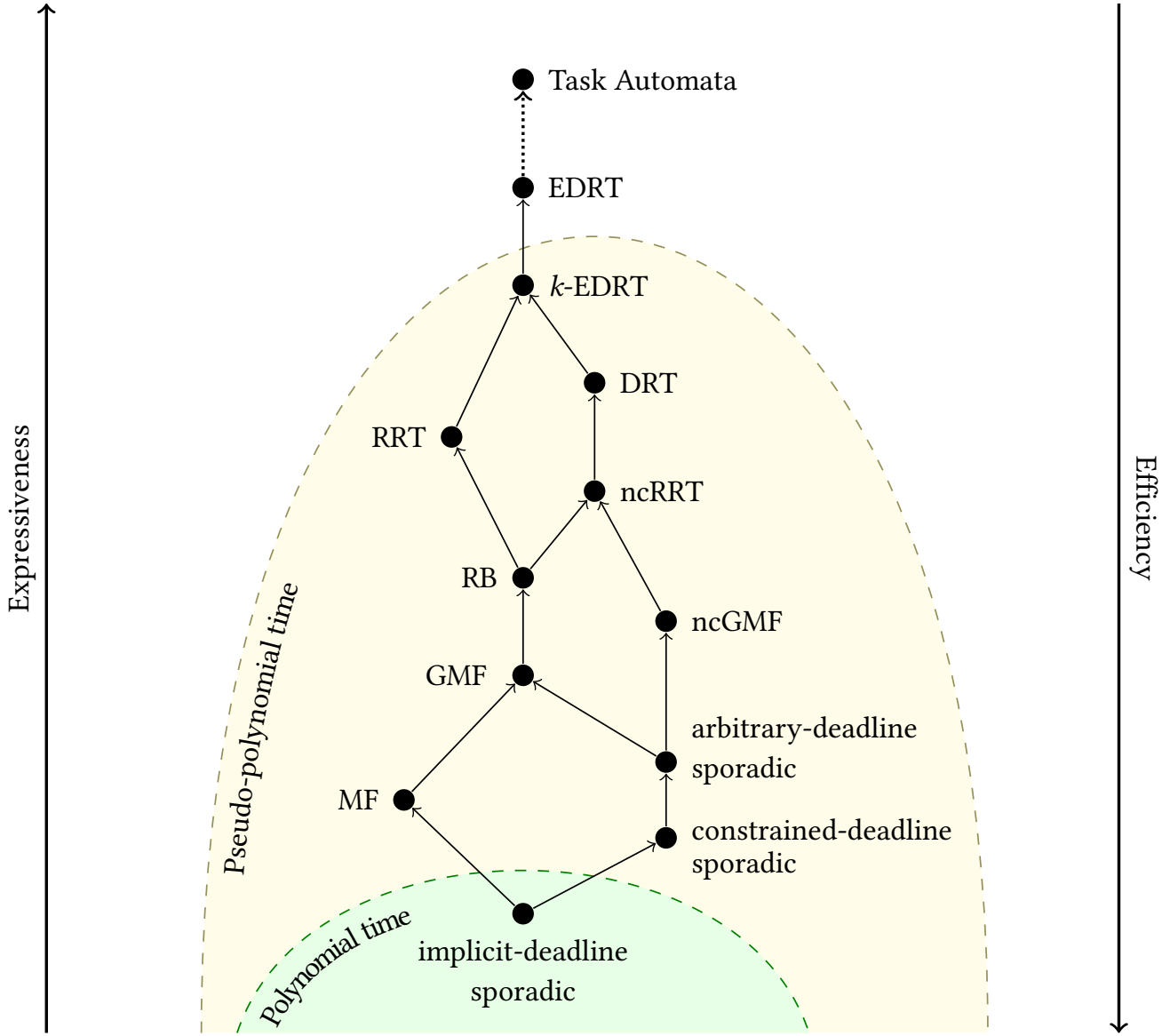


Pushing the boundary of pseudo-polynomial time



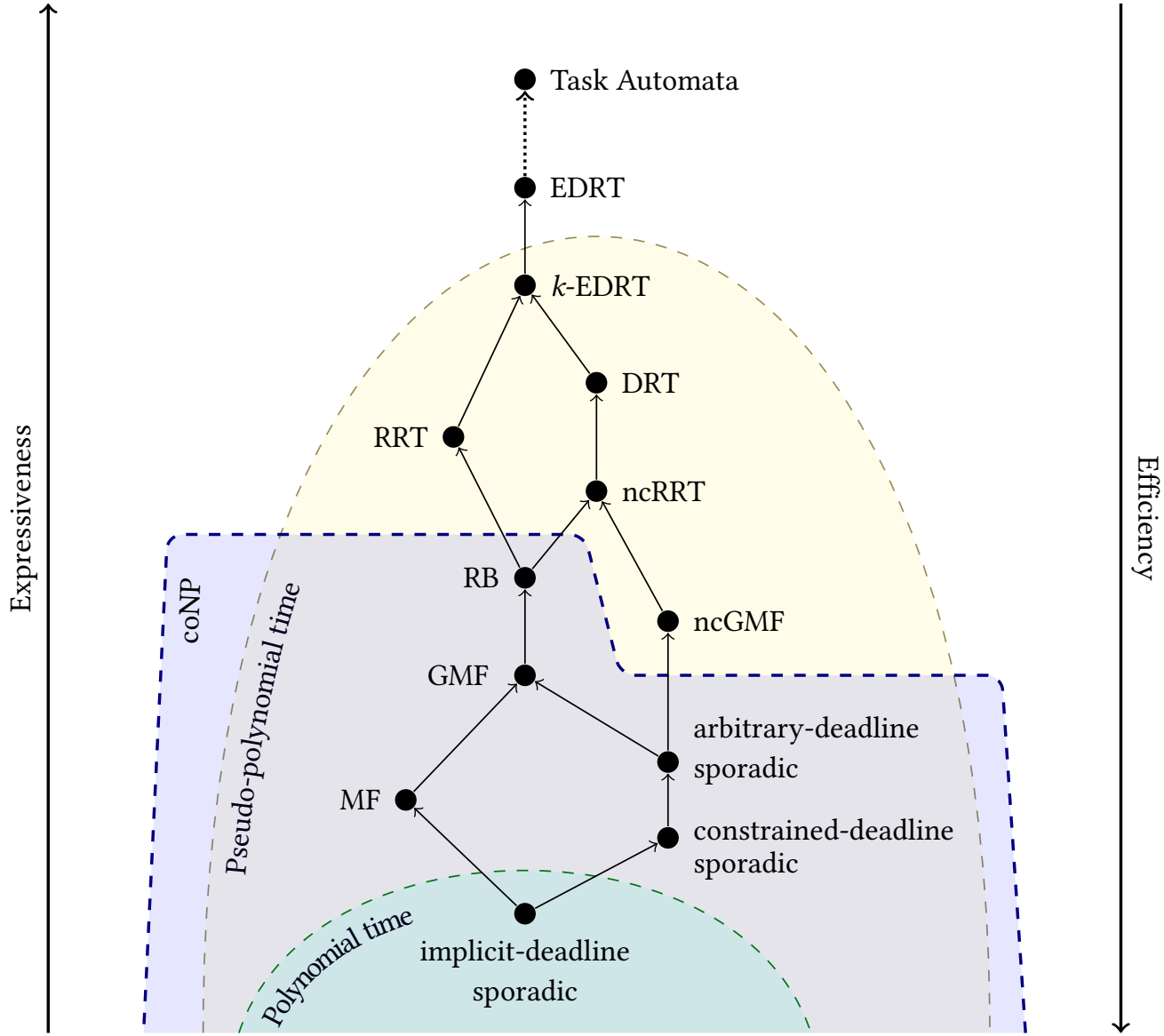
(Single processor Earliest Deadline First scheduling)

Pushing the boundary of pseudo-polynomial time



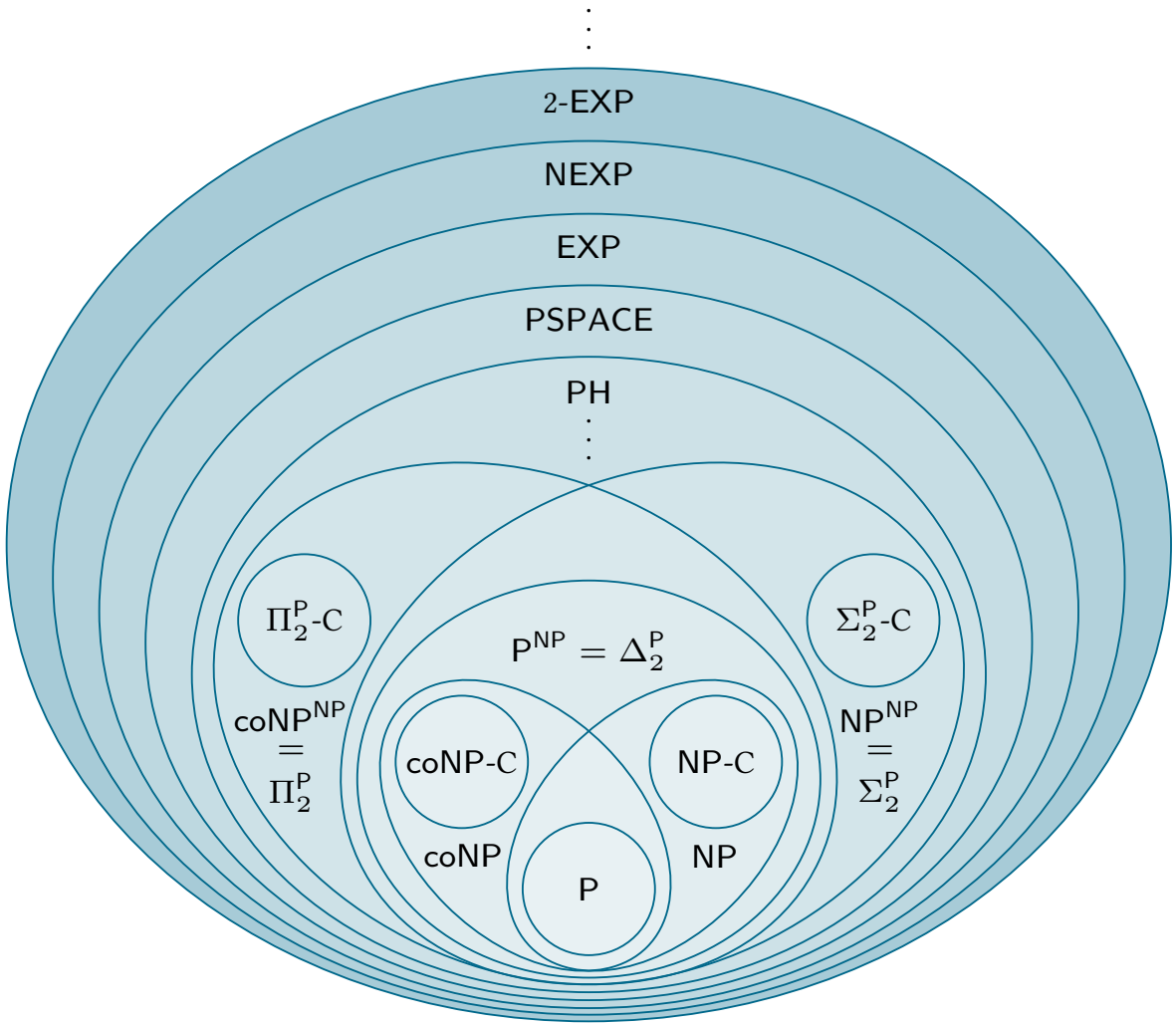
(Single processor Earliest Deadline First scheduling)

Pushing the boundary of pseudo-polynomial time

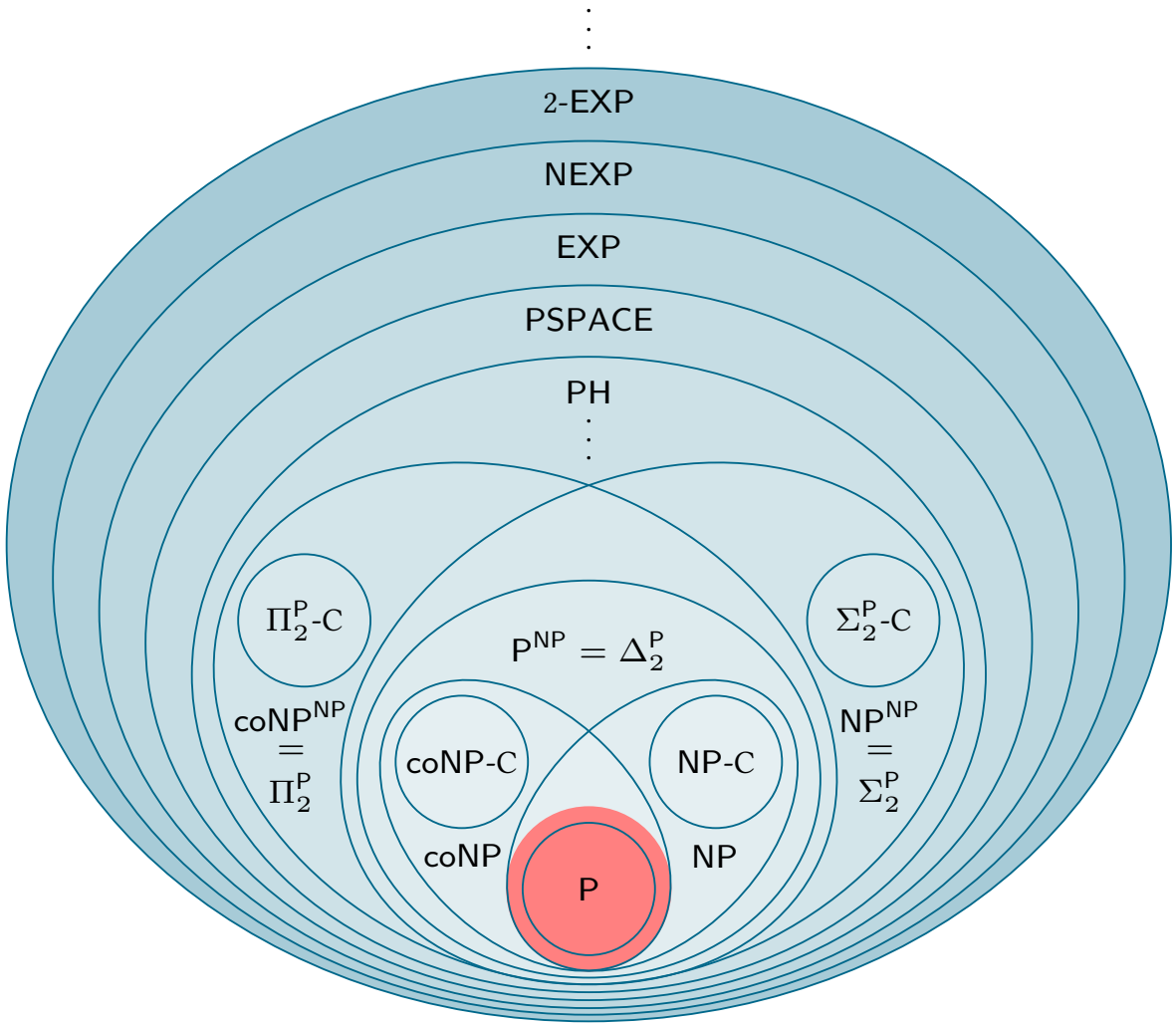


(Single processor Earliest Deadline First scheduling)

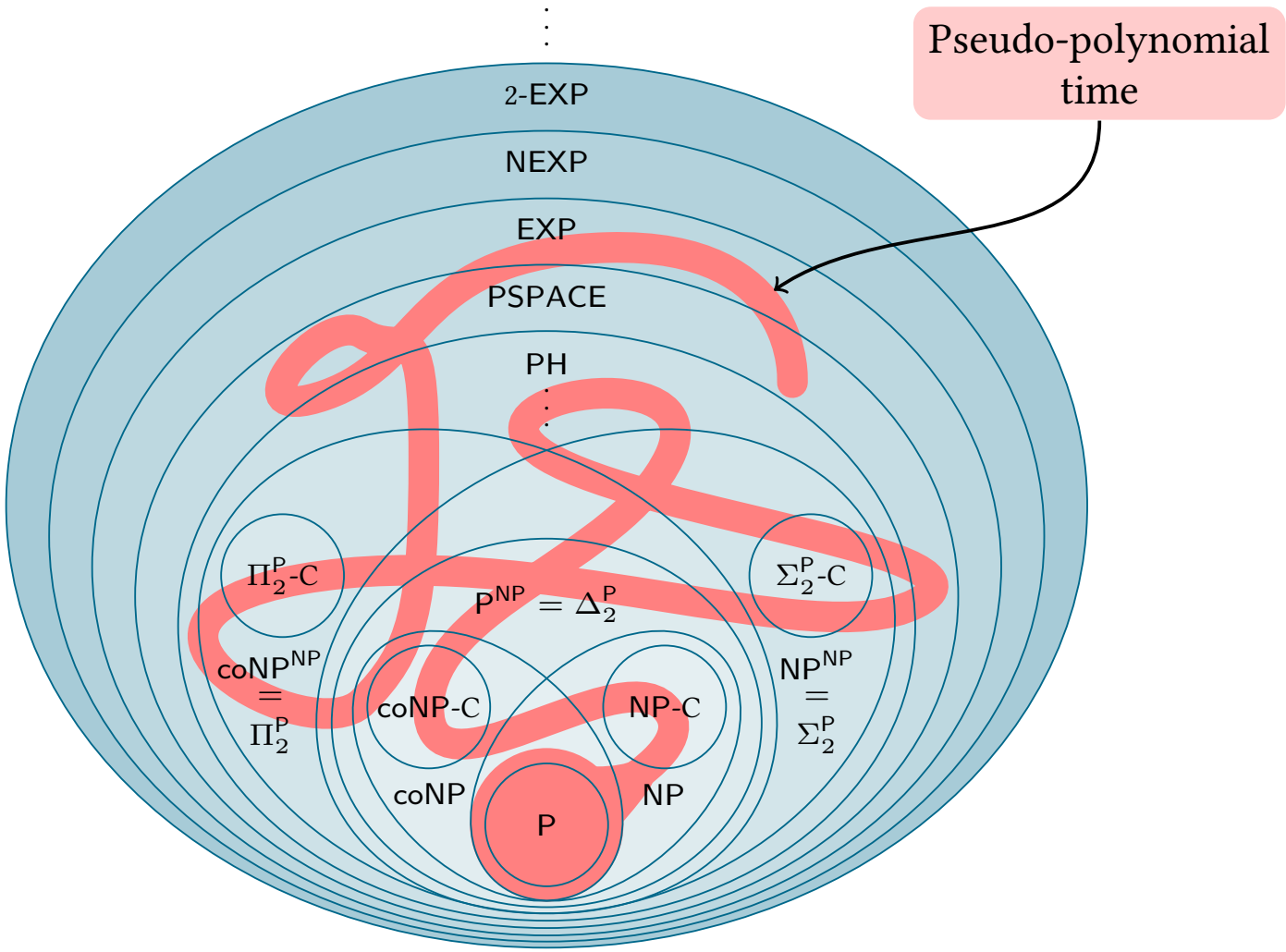
Where is pseudo-polynomial time?



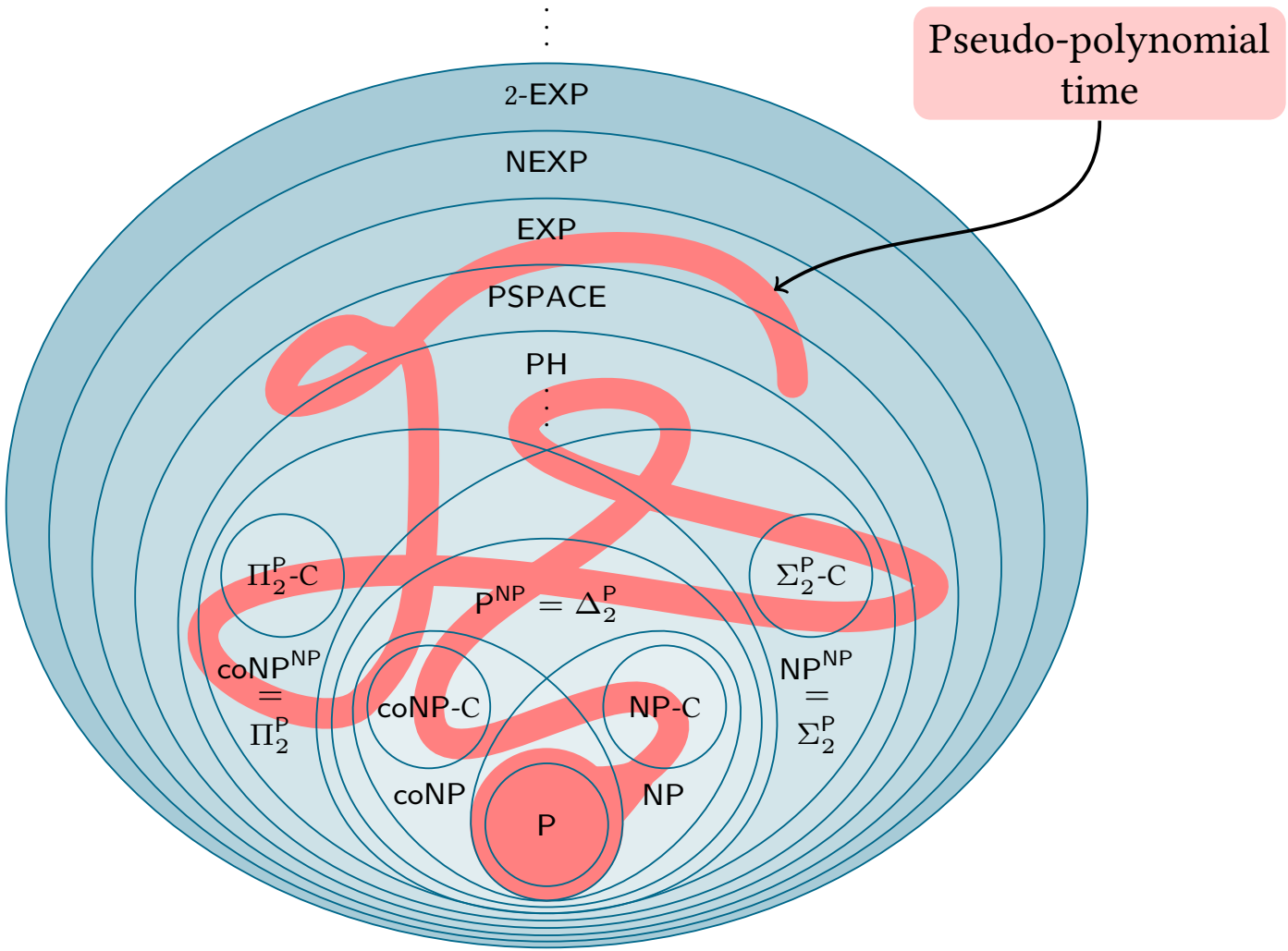
Where is pseudo-polynomial time?



Where is pseudo-polynomial time?



Where is pseudo-polynomial time?



Every C-complete class has pseudo-polynomial problems if $C \subseteq EXP$ and C is closed under polynomial-time reductions.

A more fine-grained take on pseudo-polynomial time

Pseudo-polynomial time: $\mathsf{poly}(n, N)$

A more fine-grained take on pseudo-polynomial time

Pseudo-polynomial time: $\mathsf{poly}(n, N)$

But even for "nice" problems, we expect $N \gg n$.

A more fine-grained take on pseudo-polynomial time

Pseudo-polynomial time: $\mathsf{poly}(n, N)$

But even for "nice" problems, we expect $N \gg n$.

Definition

- An algorithm is *pseudo-linear* if it is $\mathcal{O}(n^k \times N)$.
- An algorithm is *pseudo-quadratic* if it is $\mathcal{O}(n^k \times N^2)$.

A more fine-grained take on pseudo-polynomial time

Pseudo-polynomial time: $\mathsf{poly}(n, N)$

But even for "nice" problems, we expect $N \gg n$.

Definition

- An algorithm is *pseudo-linear* if it is $\mathcal{O}(n^k \times N)$.
- An algorithm is *pseudo-quadratic* if it is $\mathcal{O}(n^k \times N^2)$.

  ...
- An algorithm is *pseudo-f* if it is $\mathcal{O}(n^k \times f(N))$.

A more fine-grained take on pseudo-polynomial time

Pseudo-polynomial time: $\mathsf{poly}(n, N)$

But even for "nice" problems, we expect $N \gg n$.

Definition

- An algorithm is *pseudo-linear* if it is $\mathcal{O}(n^k \times N)$.
- An algorithm is *pseudo-quadratic* if it is $\mathcal{O}(n^k \times N^2)$.

  ...
- An algorithm is *pseudo-f* if it is $\mathcal{O}(n^k \times f(N))$.

$\mathsf{TIME}(\mathsf{poly}(n) \times N^a) \subset \mathsf{TIME}(\mathsf{poly}(n) \times N^b)$, if $0 < a < b$

A more fine-grained take on pseudo-polynomial time

Pseudo-polynomial time: $\mathsf{poly}(n, N)$

But even for "nice" problems, we expect $N \gg n$.

Definition

- An algorithm is *pseudo-linear* if it is $\mathcal{O}(n^k \times N)$.
- An algorithm is *pseudo-quadratic* if it is $\mathcal{O}(n^k \times N^2)$.

  ...
- An algorithm is *pseudo-f* if it is $\mathcal{O}(n^k \times f(N))$.

$$\mathsf{TIME}(\mathsf{poly}(n) \times N^a) \subset \mathsf{TIME}(\mathsf{poly}(n) \times N^b), \text{ if } 0 < a < b$$

$$\text{If } f(N) = g(N) \times \log^k(N), \text{ then pseudo-}f = \text{pseudo-}g$$

A few examples

A few examples

Response time analysis (RTA)

Processor demand analysis (PDA)

A few examples

Response time analysis (RTA)

pseudo-linear

Processor demand analysis (PDA)

pseudo-linear

A few examples

Response time analysis (RTA)

pseudo-linear

Processor demand analysis (PDA)

pseudo-linear

EDF DRT analysis

EDF DRT analysis (arbitrary deadlines)

A few examples

Response time analysis (RTA)

pseudo-linear

Processor demand analysis (PDA)

pseudo-linear

EDF DRT analysis

EDF DRT analysis (arbitrary deadlines)

A few examples

Response time analysis (RTA)

pseudo-linear

Processor demand analysis (PDA)

pseudo-linear

EDF DRT analysis

pseudo-quadratic

EDF DRT analysis (arbitrary deadlines)

A few examples

Response time analysis (RTA)

pseudo-linear

Processor demand analysis (PDA)

pseudo-linear

EDF DRT analysis

pseudo-quadratic

EDF DRT analysis (arbitrary deadlines)

pseudo-cubic

A few examples

Response time analysis (RTA)

pseudo-linear

Processor demand analysis (PDA)

pseudo-linear

EDF DRT analysis

~~*pseudo-quadratic*~~

pseudo-linear

(with optimization trick)

EDF DRT analysis (arbitrary deadlines)

pseudo-cubic

A few examples

Response time analysis (RTA)

pseudo-linear

Processor demand analysis (PDA)

pseudo-linear

EDF DRT analysis

~~*pseudo-quadratic*~~

pseudo-linear

(with optimization trick)

EDF DRT analysis (arbitrary deadlines)

~~*pseudo-cubic*~~

pseudo-quadratic

(with optimization trick)

A more fine-grained take on pseudo-polynomial time

A more fine-grained take on pseudo-polynomial time

Changing time units should preferably not change running time.

A more fine-grained take on pseudo-polynomial time

Changing time units should preferably not change running time.

$\Downarrow$

Running times should be $\mathsf{poly}(n, N/G)$, where
G is the GCD of the numerical parameters.

A more fine-grained take on pseudo-polynomial time

Changing time units should preferably not change running time.

$$\Downarrow$$

Running times should be $\mathsf{poly}(n, N/G)$, where
G is the GCD of the numerical parameters.

Definition

A running time of $\mathsf{poly}(n, N/G)$ is *scale invariant*.

A more fine-grained take on pseudo-polynomial time

Changing time units should preferably not change running time.

$$\Downarrow$$

Running times should be $\mathsf{poly}(n, N/G)$, where
G is the GCD of the numerical parameters.

Definition

A running time of $\mathsf{poly}(n, N/G)$ is *scale invariant*.

Not all pseudo-polynomial time problems are scale-invariant.

A few examples again

A few examples again

Response time analysis (RTA)

Processor demand analysis (PDA)

A few examples again

Response time analysis (RTA)

scale invariant

Processor demand analysis (PDA)

A few examples again

Response time analysis (RTA)

scale invariant

Processor demand analysis (PDA)

not scale invariant

A few examples again

Response time analysis (RTA)

scale invariant

Processor demand analysis (PDA)

~~*not scale invariant*~~

scale invariant
(with optimization trick)

A few examples again

Response time analysis (RTA)

scale invariant

Processor demand analysis (PDA)

~~*not scale invariant*~~

scale invariant
(with optimization trick)

EDF DRT analysis

A few examples again

Response time analysis (RTA)

scale invariant

Processor demand analysis (PDA)

~~*not scale invariant*~~

scale invariant
(with optimization trick)

EDF DRT analysis

???

A few examples again

Response time analysis (RTA)

scale invariant

Processor demand analysis (PDA)

~~*not scale invariant*~~

scale invariant
(with optimization trick)

EDF DRT analysis

~~*???*~~

scale invariant
(with generic trick)

A few examples again

Response time analysis (RTA)

scale invariant

Processor demand analysis (PDA)

~~*not scale invariant*~~

scale invariant
(with optimization trick)

EDF DRT analysis

~~*???*~~

scale invariant
(with generic trick)

But several others seem to be

not scale invariant

Some takeaways

Some takeaways

1. Pseudo-polynomial problems can be found in unexpected places.

Some takeaways

1. Pseudo-polynomial problems can be found in unexpected places.
2. There is plenty of structure inside pseudo-polynomial time.

Some takeaways

1. Pseudo-polynomial problems can be found in unexpected places.
2. There is plenty of structure inside pseudo-polynomial time.
   - Significant practical consequences that should not be ignored.

Some takeaways

1. Pseudo-polynomial problems can be found in unexpected places.
2. There is plenty of structure inside pseudo-polynomial time.
   - Significant practical consequences that should not be ignored.
   - New and interesting classification tasks!

What was this really about?

Pseudo-Intellectual Ravings on Pseudo-Polynomial Time

Sanjoy Baruah[1] and Pontus Ekberg[2]

[1] Washington University in St. Louis <baruah@wustl.edu>
[2] Uppsala University <pontus.ekberg@it.uu.se>

Abstract. Mention Wang's role in schedulability analysis – pushing the boundaries of pseudo-polynomial time

Keywords: First keyword · Second keyword · Another keyword.

1 Introduction

The **schedulability analysis** problem in one of the foundational problems studied in real-time scheduling theory; it may be described as follows:

> GIVEN (i) the specifications of the computational demands of, and the timing constraints upon, a workload; (ii) the platform upon which this workload is to be implemented; and (iii) the run-time resource allocation and scheduling

$\forall$Thank you!

$\exists$Questions?