Fixed-Parameter Analysis of Preemptive Uniprocessor Scheduling Problems

Sanjoy Baruah
Washington University in Saint Louis

Pontus Ekberg
Uppsala University

Abhishek Singh
Washington University in Saint Louis

RTSS 2022

What?

What is *fixed-parameter analysis* (or parameterized complexity)?

What?

What is *fixed-parameter analysis* (or parameterized complexity)?

In short

Complexity as a function of both the *input size* and a problem-specific *parameter*.

What?

What is *fixed-parameter analysis* (or parameterized complexity)?

In short

Complexity as a function of both the *input size* and a problem-specific *parameter*.

But, why?

Many intractable problems are tractable when the right parameters are kept "small".

What?

What is *fixed-parameter analysis* (or parameterized complexity)?

In short

Complexity as a function of both the *input size* and a problem-specific *parameter*.

But, why?

Many intractable problems are tractable when the right parameters are kept "small".

Popularized by Downey and Fellows from the '90s.

Example

A familiar decision problem

Input: A constrained-deadline sporadic task set $\mathcal{T}$.

Question: Is $\mathcal{T}$ FP-schedulable on a single processor?

Example

A familiar decision problem

Input: A constrained-deadline sporadic task set $\mathcal{T}$.

Question: Is $\mathcal{T}$ FP-schedulable on a single processor?

This problem is NP-complete!

Example

A familiar decision problem

Input: A constrained-deadline sporadic task set $\mathcal{T}$.

Question: Is $\mathcal{T}$ FP-schedulable on a single processor?

This problem is NP-complete!

$$\Longrightarrow$$

If $\mathsf{P} \neq \mathsf{NP}$, there is no algorithm to solve it with runtime $\text{poly}(n)$, where n is the *size of the input* (#bits needed to represent $\mathcal{T}$)

Example

A familiar decision problem

Input: A constrained-deadline sporadic task set $\mathcal{T}$.

Question: Is $\mathcal{T}$ FP-schedulable on a single processor?

This problem is NP-complete!

$\Longrightarrow$

If $\mathsf{P} \neq \mathsf{NP}$, there is no algorithm to solve it with runtime $\text{poly}(n)$, where n is the *size of the input* (#bits needed to represent $\mathcal{T}$)

?

What if the runtime is expressed as a function of both the input size n and a parameter k?

Example

A familiar decision problem

Input: A constrained-deadline sporadic task set $\mathcal{T}$.

Question: Is $\mathcal{T}$ FP-schedulable on a single processor?

Parameterization 1

$k_1 =$ max numerical value in $\mathcal{T}$

Example

A familiar decision problem

Input: A constrained-deadline sporadic task set $\mathcal{T}$.

Question: Is $\mathcal{T}$ FP-schedulable on a single processor?

Parameterization 1

$k_1 =$ max numerical value in $\mathcal{T}$

Parameterization 2

$k_2 =$ number of tasks in $\mathcal{T}$

Example

A familiar decision problem

Input: A constrained-deadline sporadic task set $\mathcal{T}$.

Question: Is $\mathcal{T}$ FP-schedulable on a single processor?

Parameterization 1

$k_1 =$ max numerical value in $\mathcal{T}$

Parameterization 2

$k_2 =$ number of tasks in $\mathcal{T}$

?

Is FP-schedulability "tractable" when k_1 or k_2 are small?

Fixed-Parameter Tractable (FPT)

FPT is the class of “tractable” parameterized problems.

Fixed-Parameter Tractable (FPT)

FPT is the class of "tractable" parameterized problems.

An *fpt-algorithm* runs in time $O\big(f(k) \times \mathsf{poly}(n)\big)$.

n — size of the input

k — parameter

f — computable function

Fixed-Parameter Tractable (FPT)

FPT is the class of "tractable" parameterized problems.

An *fpt-algorithm* runs in time $O\big(f(k) \times \mathsf{poly}(n)\big)$.

- n — size of the input
- k — parameter
- f — computable function

Parameter isolated in its own factor

Fixed-Parameter Tractable (FPT)

FPT is the class of "tractable" parameterized problems.

An *fpt-algorithm* runs in time $O\big(f(k) \times \mathsf{poly}(n)\big)$.

- n — size of the input
- k — parameter
- f — computable function

Parameter isolated in its own factor

Otherwise polynomial in input size

Fixed-Parameter Tractable (FPT)

FPT is the class of "tractable" parameterized problems.

An *fpt-algorithm* runs in time $O\big(f(k) \times \mathsf{poly}(n)\big)$.

- n — size of the input
- k — parameter
- f — computable function

Otherwise polynomial in input size

Parameter isolated in its own factor

Parameterizing the FP-schedulability problem

A familiar decision problem

Input: A constrained-deadline sporadic task set $\mathcal{T}$.

Question: Is $\mathcal{T}$ FP-schedulable on a single processor?

$k_1 =$ max numerical value in $\mathcal{T}$

$k_2 =$ number of tasks in $\mathcal{T}$

Parameterizing the FP-schedulability problem

A familiar decision problem

Input: A constrained-deadline sporadic task set $\mathcal{T}$.

Question: Is $\mathcal{T}$ FP-schedulable on a single processor?

$k_1 =$ max numerical value in $\mathcal{T}$

?

In FPT?

$O\big(f(k_1) \times \mathsf{poly}(n)\big)$?

$k_2 =$ number of tasks in $\mathcal{T}$

Parameterizing the FP-schedulability problem

A familiar decision problem

Input: A constrained-deadline sporadic task set $\mathcal{T}$.

Question: Is $\mathcal{T}$ FP-schedulable on a single processor?

$k_1 =$ max numerical value in $\mathcal{T}$

?

In FPT?

$O\big(f(k_1) \times \mathsf{poly}(n)\big)$?

Yes!

$O\big(k_1 \times n^2\big)$ using RTA

$k_2 =$ number of tasks in $\mathcal{T}$

Parameterizing the FP-schedulability problem

A familiar decision problem

Input: A constrained-deadline sporadic task set $\mathcal{T}$.

Question: Is $\mathcal{T}$ FP-schedulable on a single processor?

$k_1 =$ max numerical value in $\mathcal{T}$

?

In FPT?

$O\big(f(k_1) \times \mathsf{poly}(n)\big)$?

Yes!

$O\big(k_1 \times n^2\big)$ using RTA

$k_2 =$ number of tasks in $\mathcal{T}$

?

In FPT?

$O\big(f(k_2) \times \mathsf{poly}(n)\big)$?

Response-Time Analysis (RTA)

	C_i	D_i	T_i
τ_1	$x-1$	x	x
τ_2	x	x^2	x^2

Response-Time Analysis (RTA)

	C_i	D_i	T_i
τ_1	$x-1$	x	x
τ_2	x	x^2	x^2

$$R_i^{(k+1)} = C_i + \sum_{j \in \mathrm{hp}(i)} \left\lceil \frac{R_i^{(k)}}{T_j} \right\rceil \times C_j$$

Response-Time Analysis (RTA)

	C_i	D_i	T_i
τ_1	$x-1$	x	x
τ_2	x	x^2	x^2

$$R_i^{(k+1)} = C_i + \sum_{j \in \mathrm{hp}(i)} \left\lceil \frac{R_i^{(k)}}{T_j} \right\rceil \times C_j$$

Response-Time Analysis (RTA)

	C_i	D_i	T_i
τ_1	$x-1$	x	x
τ_2	x	x^2	x^2

$$R_i^{(k+1)} = C_i + \sum_{j \in \mathrm{hp}(i)} \left\lceil \frac{R_i^{(k)}}{T_j} \right\rceil \times C_j$$

$$R_2^{(0)} = 2x - 1$$

$$R_2^{(1)} = 3x - 2$$

$$R_2^{(2)} = 4x - 3$$

$$\vdots$$

$$R_2^{(x-1)} = x^2$$

Response-Time Analysis (RTA)

	C_i	D_i	T_i
τ_1	$x-1$	x	x
τ_2	x	x^2	x^2

$$R_i^{(k+1)} = C_i + \sum_{j \in \mathrm{hp}(i)} \left\lceil \frac{R_i^{(k)}}{T_j} \right\rceil \times C_j$$

$$\left.\begin{aligned} R_2^{(0)} &= 2x-1 \\ R_2^{(1)} &= 3x-2 \\ R_2^{(2)} &= 4x-3 \\ &\vdots \\ R_2^{(x-1)} &= x^2 \end{aligned}\right\} \; x \text{ iterations!}$$

Response-Time Analysis (RTA)

	C_i	D_i	T_i
τ_1	$x-1$	x	x
τ_2	x	x^2	x^2

$$R_i^{(k+1)} = C_i + \sum_{j \in \mathrm{hp}(i)} \left\lceil \frac{R_i^{(k)}}{T_j} \right\rceil \times C_j$$

RTA is *not*
$O\big(f(\#tasks) \times \mathsf{poly}(n)\big)$
and is therefore *not*
an fpt-algorithm here

$$\left.\begin{aligned} R_2^{(0)} &= 2x-1 \\ R_2^{(1)} &= 3x-2 \\ R_2^{(2)} &= 4x-3 \\ &\vdots \\ R_2^{(x-1)} &= x^2 \end{aligned}\right\} \; x \text{ iterations!}$$

Parameterizing the FP-schedulability problem

A familiar decision problem

Input: A constrained-deadline sporadic task set $\mathcal{T}$.

Question: Is $\mathcal{T}$ FP-schedulable on a single processor?

$k_1 =$ max numerical value in $\mathcal{T}$

?

In FPT?

$O\big(f(k_1) \times \mathsf{poly}(n)\big)$?

Yes!

$O\big(k_1 \times n^2\big)$ using RTA

$k_2 =$ number of tasks in $\mathcal{T}$

?

In FPT?

$O\big(f(k_2) \times \mathsf{poly}(n)\big)$?

Parameterizing the FP-schedulability problem

A familiar decision problem

Input: A constrained-deadline sporadic task set $\mathcal{T}$.

Question: Is $\mathcal{T}$ FP-schedulable on a single processor?

$k_1 =$ max numerical value in $\mathcal{T}$

?

In FPT?

$O\big(f(k_1) \times \text{poly}(n)\big)$?

Yes!

$O\big(k_1 \times n^2\big)$ using RTA

$k_2 =$ number of tasks in $\mathcal{T}$

?

In FPT?

$O\big(f(k_2) \times \text{poly}(n)\big)$?

Not with RTA!

Hyperplanes Exact Test (HET)

HET

Another FP-schedulability test by Bini & Buttazzo (2004).

(Similar ideas also presented by Manabe & Aoyagi (1995).)

Hyperplanes Exact Test (HET)

HET

Another FP-schedulability test by Bini & Buttazzo (2004).

(Similar ideas also presented by Manabe & Aoyagi (1995).)

HET does *not* use the iterative RTA approach.

HET directly evaluates *at most* $2^{\#tasks}$ points in the RTA equation.

Hyperplanes Exact Test (HET)

HET

Another FP-schedulability test by Bini & Buttazzo (2004).

(Similar ideas also presented by Manabe & Aoyagi (1995).)

HET does *not* use the iterative RTA approach.

HET directly evaluates *at most* $2^{\#tasks}$ points in the RTA equation.

$$\Longrightarrow$$

HET runs in $O\big(f(\#tasks) \times \mathsf{poly}(n)\big)$ time!

Parameterizing the FP-schedulability problem

A familiar decision problem

Input: A constrained-deadline sporadic task set $\mathcal{T}$.

Question: Is $\mathcal{T}$ FP-schedulable on a single processor?

$k_1 =$ max numerical value in $\mathcal{T}$

?

In FPT?

$O\big(f(k_1) \times \mathsf{poly}(n)\big)$?

Yes!

$O\big(k_1 \times n^2\big)$ using RTA

$k_2 =$ number of tasks in $\mathcal{T}$

?

In FPT?

$O\big(f(k_2) \times \mathsf{poly}(n)\big)$?

Not with RTA!

Parameterizing the FP-schedulability problem

A familiar decision problem

Input: A constrained-deadline sporadic task set $\mathcal{T}$.

Question: Is $\mathcal{T}$ FP-schedulable on a single processor?

$k_1 =$ max numerical value in $\mathcal{T}$

? In FPT?

$O\big(f(k_1) \times \mathsf{poly}(n)\big)$?

Yes!

$O\big(k_1 \times n^2\big)$ using RTA

$k_2 =$ number of tasks in $\mathcal{T}$

? In FPT?

$O\big(f(k_2) \times \mathsf{poly}(n)\big)$?

Yes!

$O\big(2^{k_2} \times n^2\big)$ using HET

Parameterizing the FP-schedulability problem

A familiar decision problem

Input: A constrained-deadline sporadic task set $\mathcal{T}$.

Question: Is $\mathcal{T}$ FP-schedulable on a single processor?

$k_1 =$ max numerical value in $\mathcal{T}$

?

In FPT?

$O\big(f(k_1) \times \text{poly}(n)\big)$?

Yes!

$O\big(k_1 \times n^2\big)$ using RTA

$k_2 =$ number of tasks in $\mathcal{T}$

?

In FPT?

$O\big(f(k_2) \times \text{poly}(n)\big)$?

Yes!

$O\big(2^{k_2} \times n^2\big)$ using HET

HET is an fpt-alg. even with

$k =$ number of distinct periods

Parameterizing the FP-schedulability problem

A familiar decision problem

Input: A constrained-deadline sporadic task set $\mathcal{T}$.

Question: Is $\mathcal{T}$ FP-schedulable on a single processor?

$k_1 =$ max numerical value in $\mathcal{T}$

?

In FPT?

$O\big(f(k_1) \times \text{poly}(n)\big)$?

Yes!

$O\big(k_1 \times n^2\big)$ using RTA

RTA is an fpt-alg. even with $k = T_{\max}/T_{\min}$

$k_2 =$ number of tasks in $\mathcal{T}$

?

In FPT?

$O\big(f(k_2) \times \text{poly}(n)\big)$?

Yes!

$O\big(2^{k_2} \times n^2\big)$ using HET

HET is an fpt-alg. even with $k =$ number of distinct periods

EDF?

Another familiar decision problem

Input: A constrained-deadline sporadic task set $\mathcal{T}$.

Question: Is $\mathcal{T}$ *EDF*-schedulable on a single processor?

EDF?

Another familiar decision problem

Input: A constrained-deadline sporadic task set $\mathcal{T}$.

Question: Is $\mathcal{T}$ *EDF*-schedulable on a single processor?

This problem is coNP-complete!

EDF?

Another familiar decision problem

Input: A constrained-deadline sporadic task set $\mathcal{T}$.

Question: Is $\mathcal{T}$ *EDF*-schedulable on a single processor?

This problem is coNP-complete!

$$k = T_{\max}/T_{\min}$$

Processor Demand Analysis (PDA) is an fpt-algorithm for *bounded-utilization* task sets

EDF?

Another familiar decision problem

Input: A constrained-deadline sporadic task set $\mathcal{T}$.

Question: Is $\mathcal{T}$ *EDF*-schedulable on a single processor?

This problem is coNP-complete!

$k = T_{\max}/T_{\min}$

Processor Demand Analysis (PDA) is an fpt-algorithm for *bounded-utilization* task sets

$k =$ number of tasks

Neither PDA nor QPA are fpt-algorithms!

EDF?

Another familiar decision problem

Input: A constrained-deadline sporadic task set $\mathcal{T}$.

Question: Is $\mathcal{T}$ *EDF*-schedulable on a single processor?

This problem is coNP-complete!

$k = T_{\max}/T_{\min}$

Processor Demand Analysis (PDA) is an fpt-algorithm for *bounded-utilization* task sets

$k =$ number of tasks

Neither PDA nor QPA are fpt-algorithms!

We can make "small" ILPs that give an fpt-algorithm, even for asynchronous tasks

To FPT, or not to FPT

?

Can we show that some problems are *not* in FPT?

To FPT, or not to FPT

?

Can we show that some problems are *not* in FPT?

$$\mathsf{FPT} \subseteq \text{para-}(\text{co})\mathsf{NP}$$

To FPT, or not to FPT

?

Can we show that some problems are *not* in FPT?

$\mathsf{FPT} \subseteq \mathsf{para\text{-}(co)NP}$ ← Strict if $\mathsf{P} \neq \mathsf{NP}$

To FPT, or not to FPT

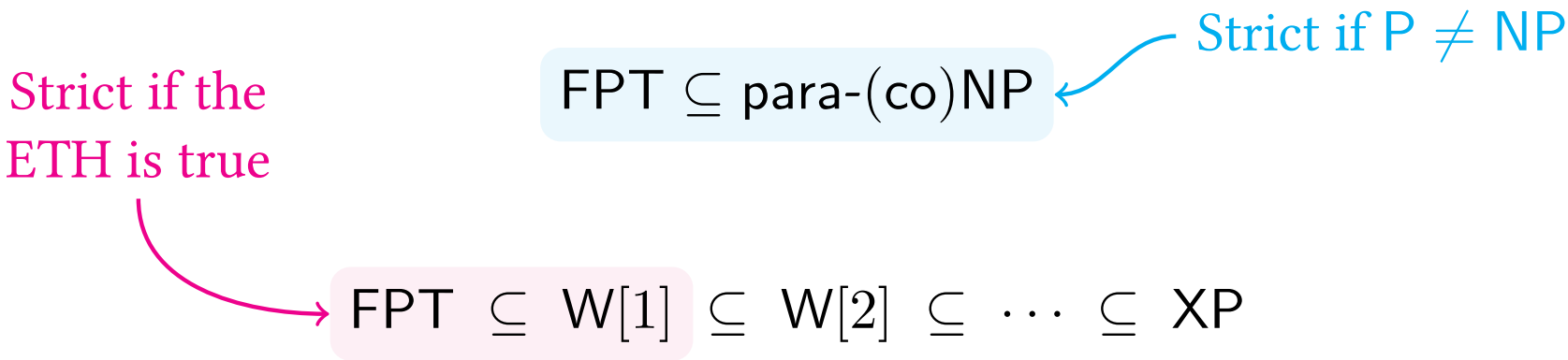
? Can we show that some problems are *not* in FPT?

$\mathsf{FPT} \subseteq \mathsf{para\text{-}(co)NP}$ ← Strict if $\mathsf{P} \neq \mathsf{NP}$

$$\mathsf{FPT} \subseteq \mathsf{W}[1] \subseteq \mathsf{W}[2] \subseteq \cdots \subseteq \mathsf{XP}$$

To FPT, or not to FPT

? Can we show that some problems are *not* in FPT?



To FPT, or not to FPT

? Can we show that some problems are *not* in FPT?

$\mathsf{FPT} \subseteq \mathsf{para\text{-}(co)NP}$ ← Strict if $\mathsf{P} \neq \mathsf{NP}$

Strict if the ETH is true →

$$\mathsf{FPT} \subseteq \mathsf{W}[1] \subseteq \mathsf{W}[2] \subseteq \cdots \subseteq \mathsf{XP}$$

Exponential Time Hypothesis (ETH)
$\approx$
3-SAT cannot be solved in sub-exponential time

Some hardness results

Some hardness results

Setting

1. Asynchronous periodic
2. Constrained deadlines
3. Work-conserving scheduler

Some hardness results

Setting

1. Asynchronous periodic
2. Constrained deadlines
3. Work-conserving scheduler

Schedulability para-coNP-hard w.

- #distinct deadlines
- #distinct WCETs
- max WCET

Some hardness results

Setting

1. Asynchronous periodic
2. Constrained deadlines
3. Work-conserving scheduler

Schedulability para-coNP-hard w.

- #distinct deadlines
- #distinct WCETs
- max WCET

Schedulability W[1]-hard w.

- max deadline

Some hardness results

Setting

1. Asynchronous periodic
2. Constrained deadlines
3. Work-conserving scheduler

Schedulability para-coNP-hard w.

- #distinct deadlines
- #distinct WCETs
- max WCET

Schedulability W[1]-hard w.

- max deadline

Setting

1. Synchronous / sporadic
2. Constrained deadlines
3. EDF

Some hardness results

Setting

1. Asynchronous periodic
2. Constrained deadlines
3. Work-conserving scheduler

Schedulability para-coNP-hard w.

- #distinct deadlines
- #distinct WCETs
- max WCET

Schedulability W[1]-hard w.

- max deadline

Setting

1. Synchronous / sporadic
2. Constrained deadlines
3. EDF

Schedulability para-coNP-hard w.

- #distinct WCETs
- max WCET

Some hardness results

Setting

1. Asynchronous periodic
2. Constrained deadlines
3. Work-conserving scheduler

Schedulability para-coNP-hard w.

- #distinct deadlines
- #distinct WCETs
- max WCET

Schedulability W[1]-hard w.

- max deadline

Setting

1. Synchronous / sporadic
2. Constrained deadlines
3. EDF

Schedulability para-coNP-hard w.

- #distinct WCETs
- max WCET

This is *not* para-coNP-hard with #distinct periods!

(Unless P = NP)

Lower bounds on f

An *fpt-algorithm* runs in time $O\big(f(k) \times \mathsf{poly}(n)\big)$.

n — size of the input

k — parameter

f — computable function

Lower bounds on f

An *fpt-algorithm* runs in time $O\big(f(k) \times \mathsf{poly}(n)\big)$.

n — size of the input

k — parameter

f — computable function

Lower bounds on f

An *fpt-algorithm* runs in time $O\big(f(k) \times \mathsf{poly}(n)\big)$.

n — size of the input

k — parameter

f — computable function

Exponential Time Hypothesis (ETH)

$\approx$

3-SAT cannot be solved in sub-exponential time

Lower bounds on f

If the ETH holds, then:

Lower bounds on f

If the ETH holds, then:

Work-conserving schedulability for *asynchronous* periodic tasks with constrained deadlines **cannot** be solved in time

$$O\left(2^{o(\#tasks)} \times \mathsf{poly}(n)\right).$$

Lower bounds on f

If the ETH holds, then:

Work-conserving schedulability for *asynchronous* periodic tasks with constrained deadlines **cannot** be solved in time

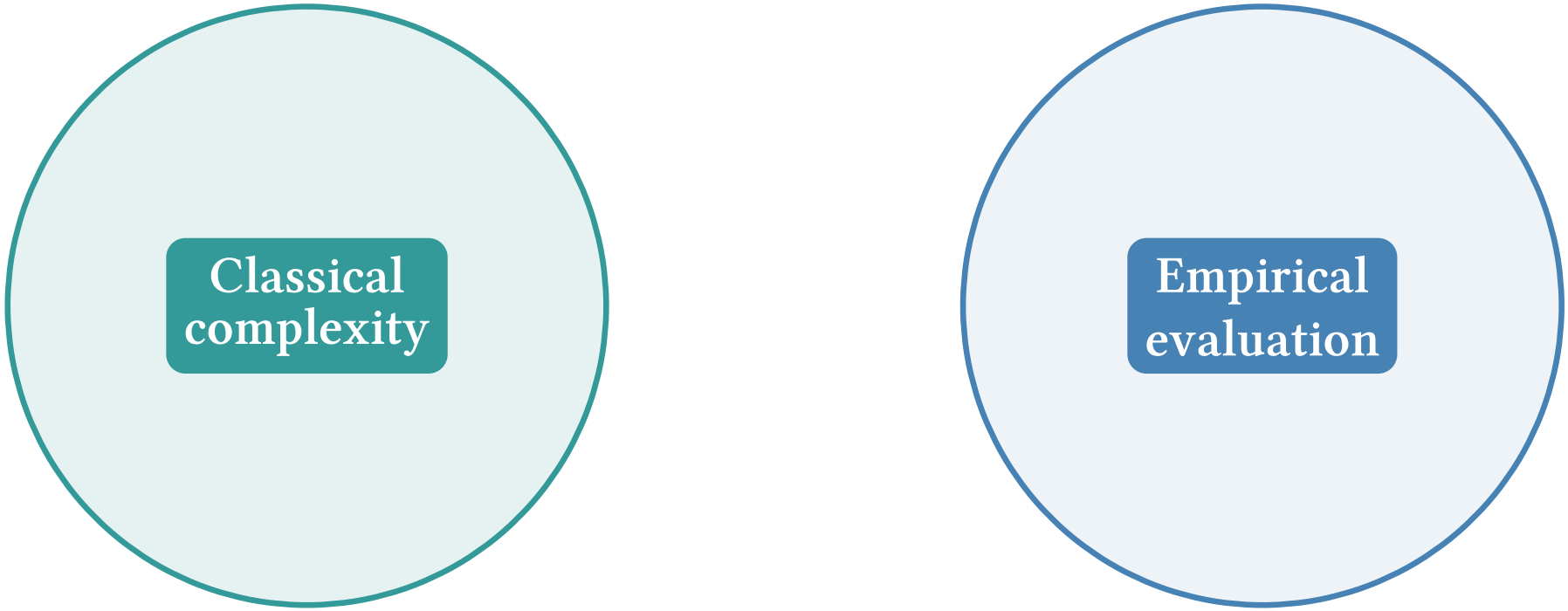
$$O\left(2^{o(\#tasks)} \times \mathsf{poly}(n)\right).$$

EDF-schedulability for *synchronous* periodic tasks with constrained deadlines **cannot** be solved in time

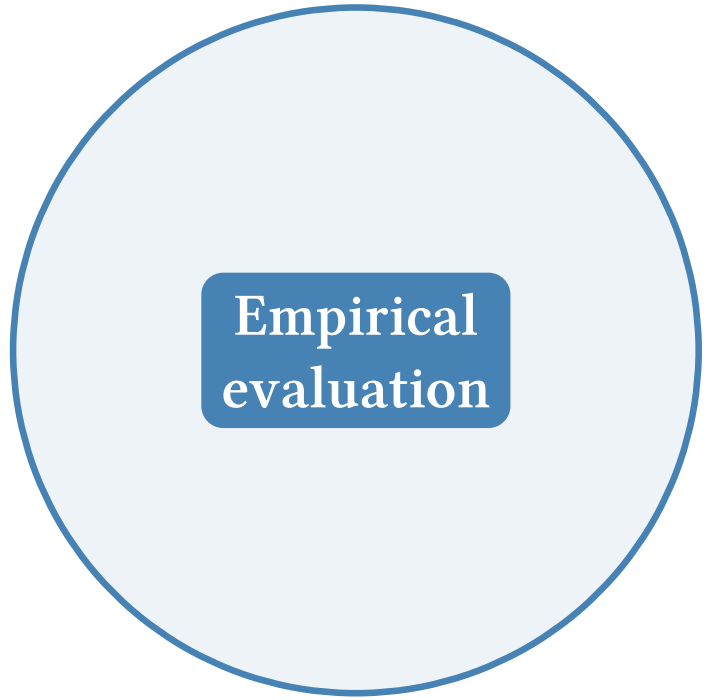
$$O\left(2^{o(\#periods)} \times \mathsf{poly}(n)\right).$$

Give us the take-home message already...

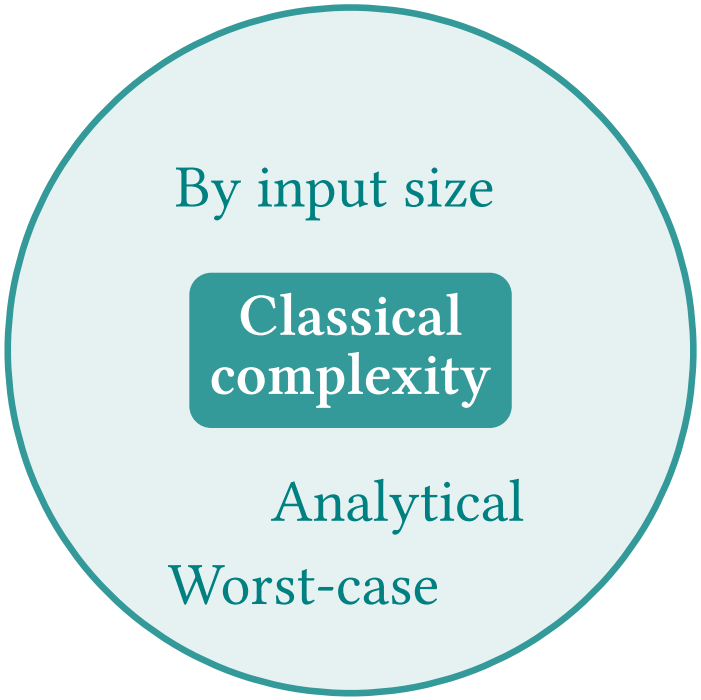
Give us the take-home message already…

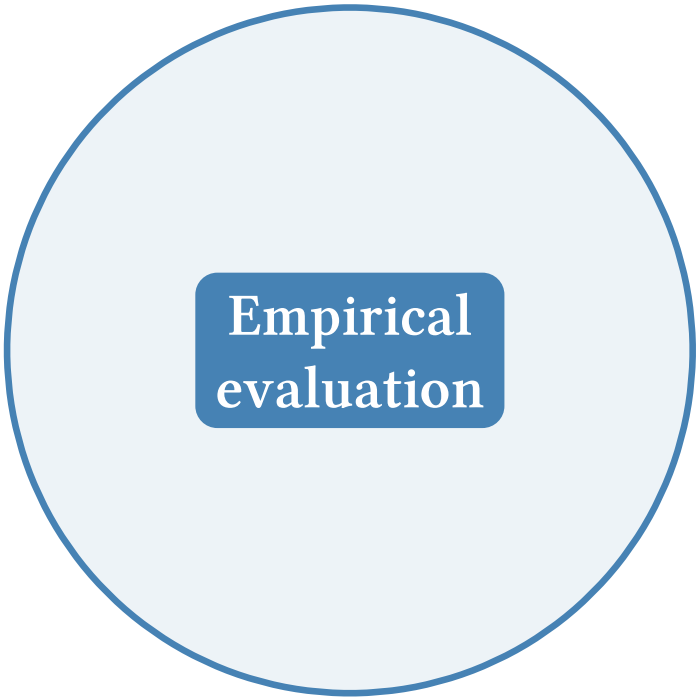


Give us the take-home message already...

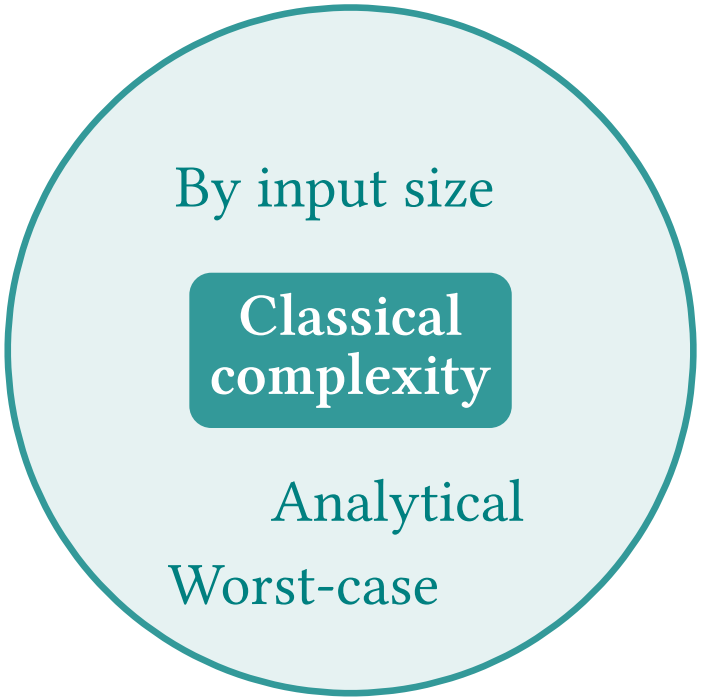


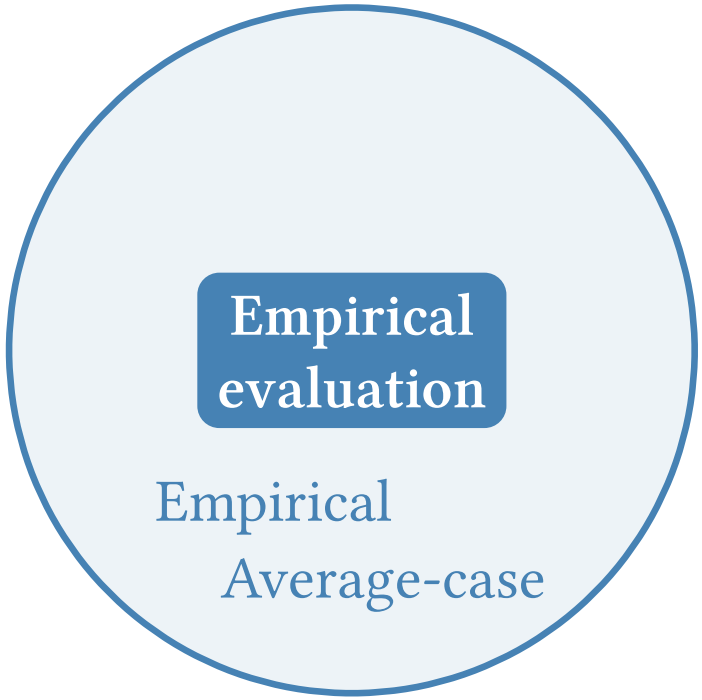
Give us the take-home message already…



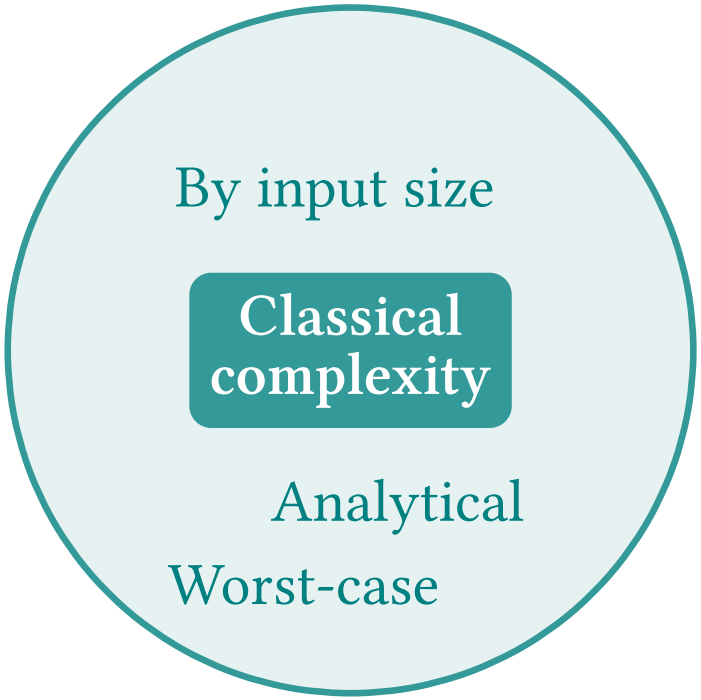


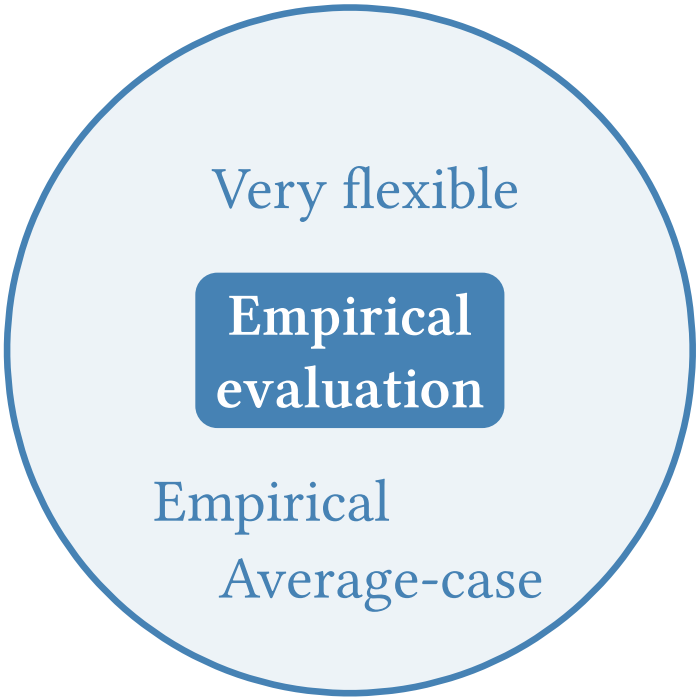
Give us the take-home message already…



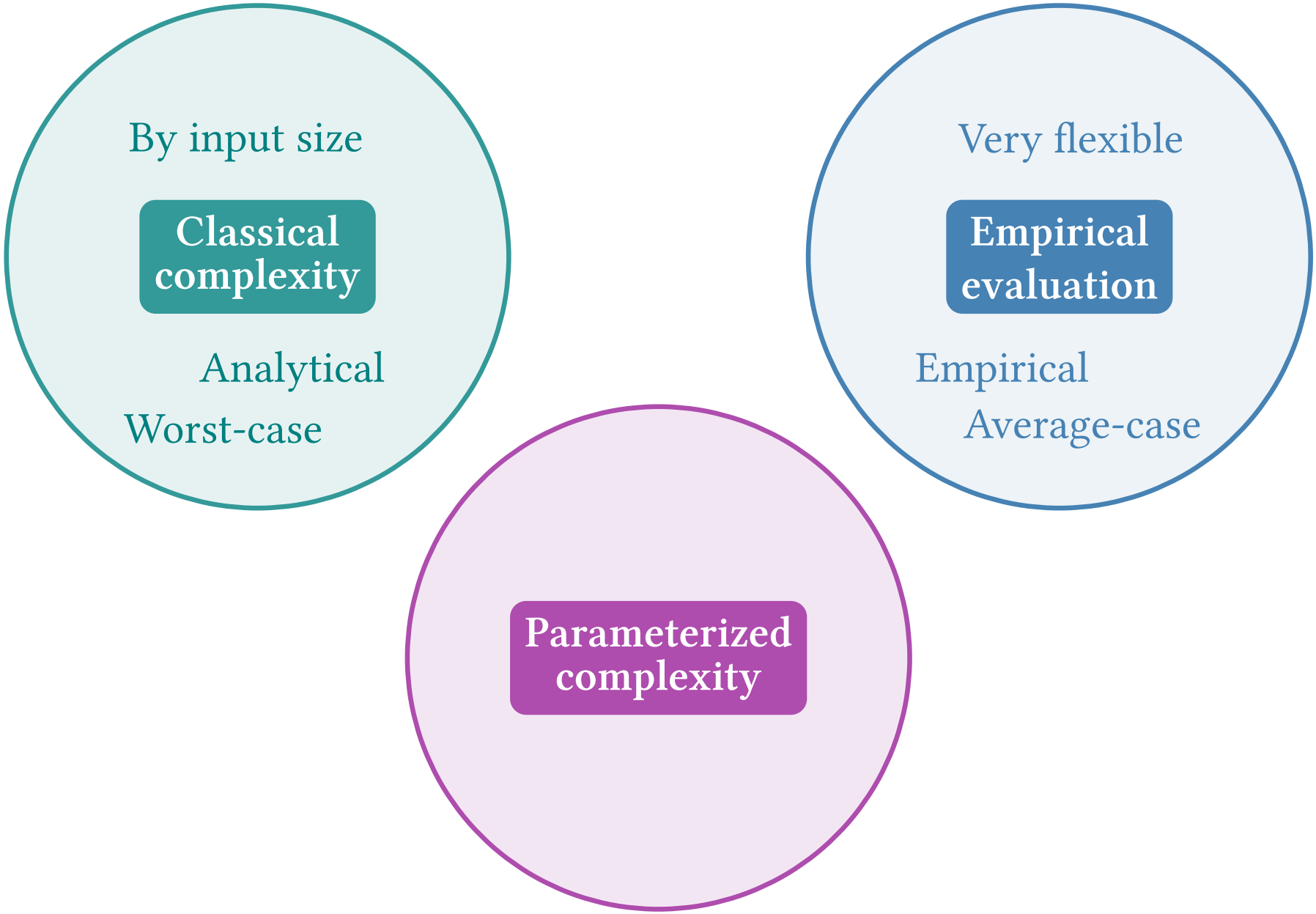


Give us the take-home message already…

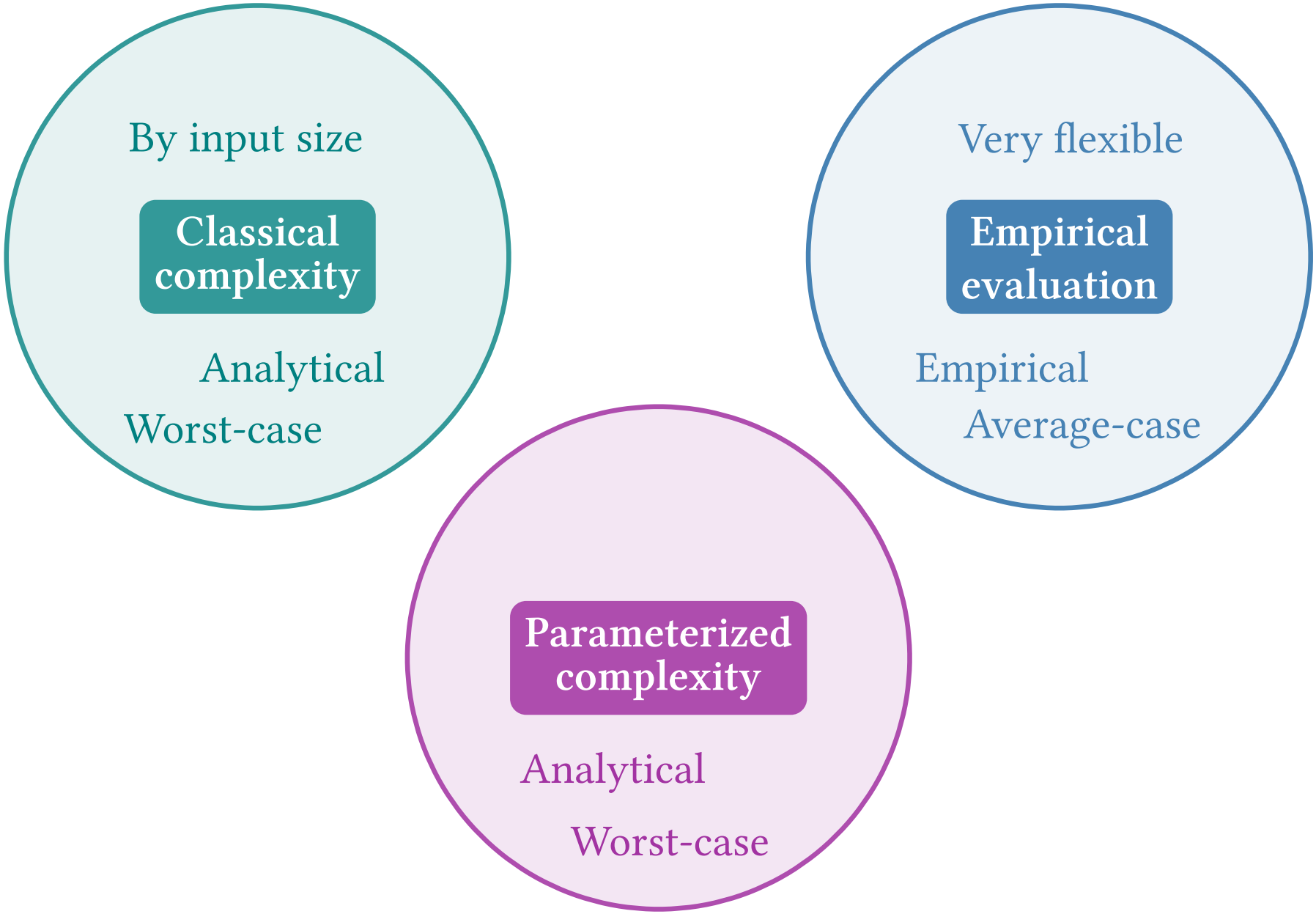




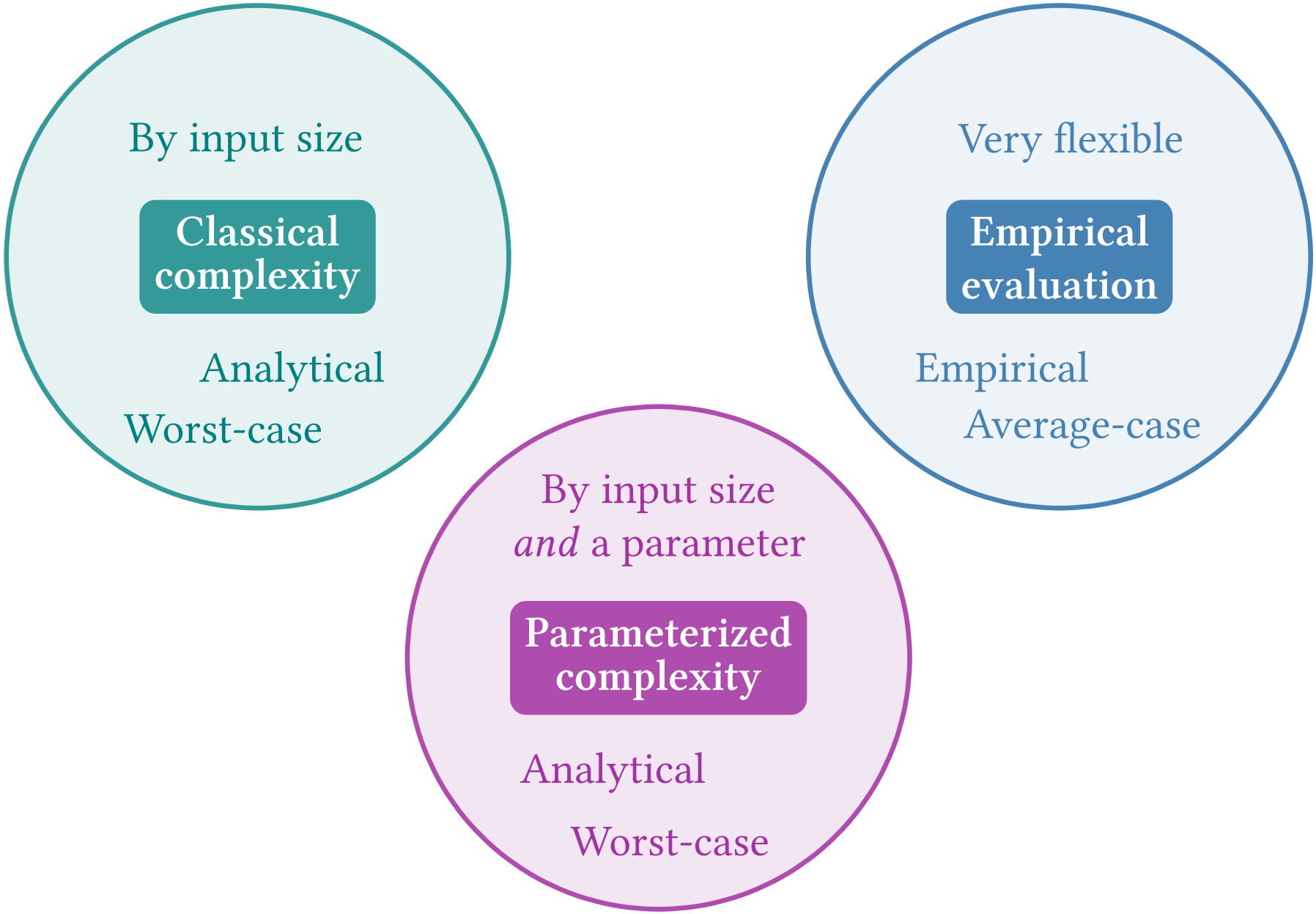
Give us the take-home message already…



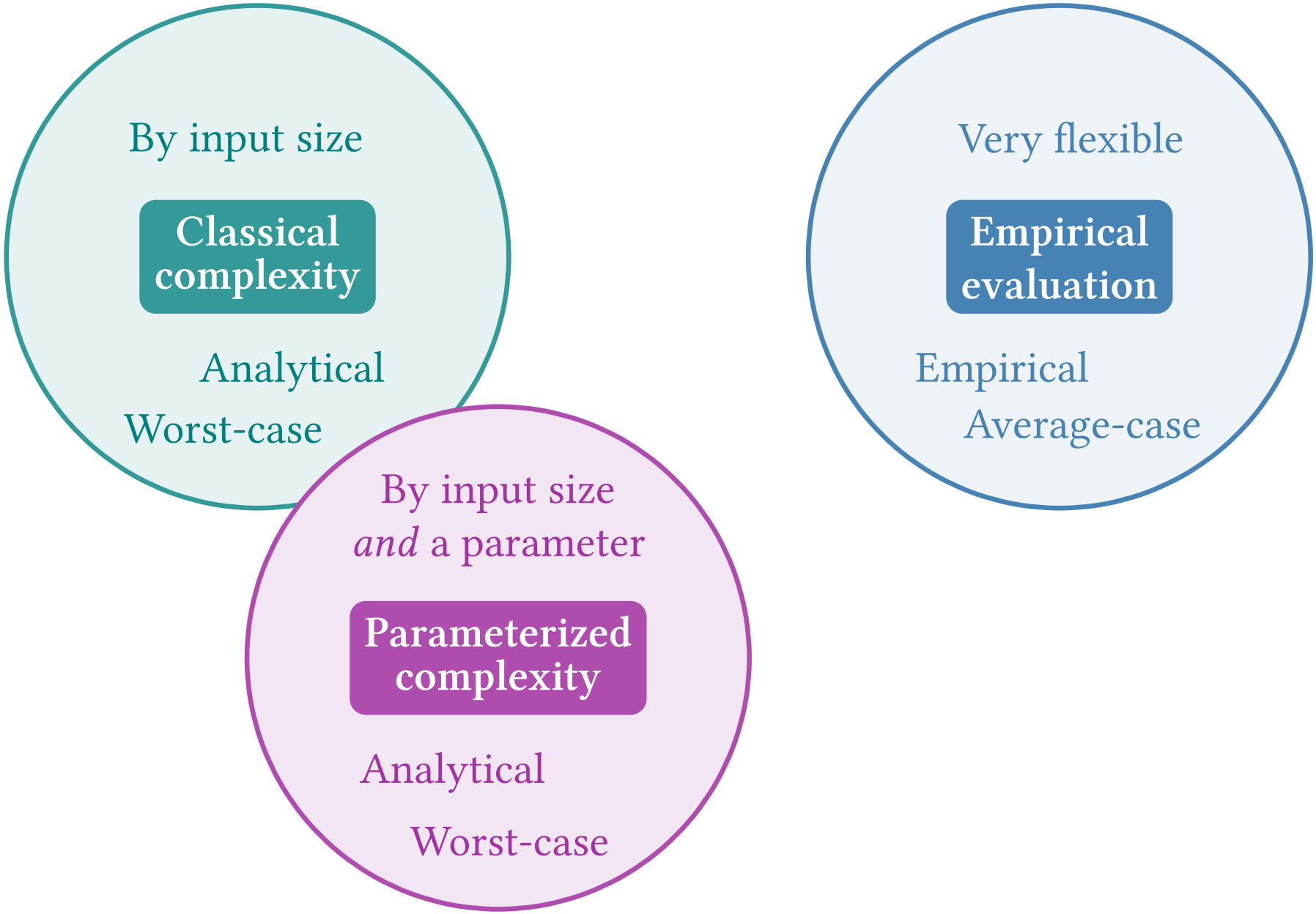
Give us the take-home message already...



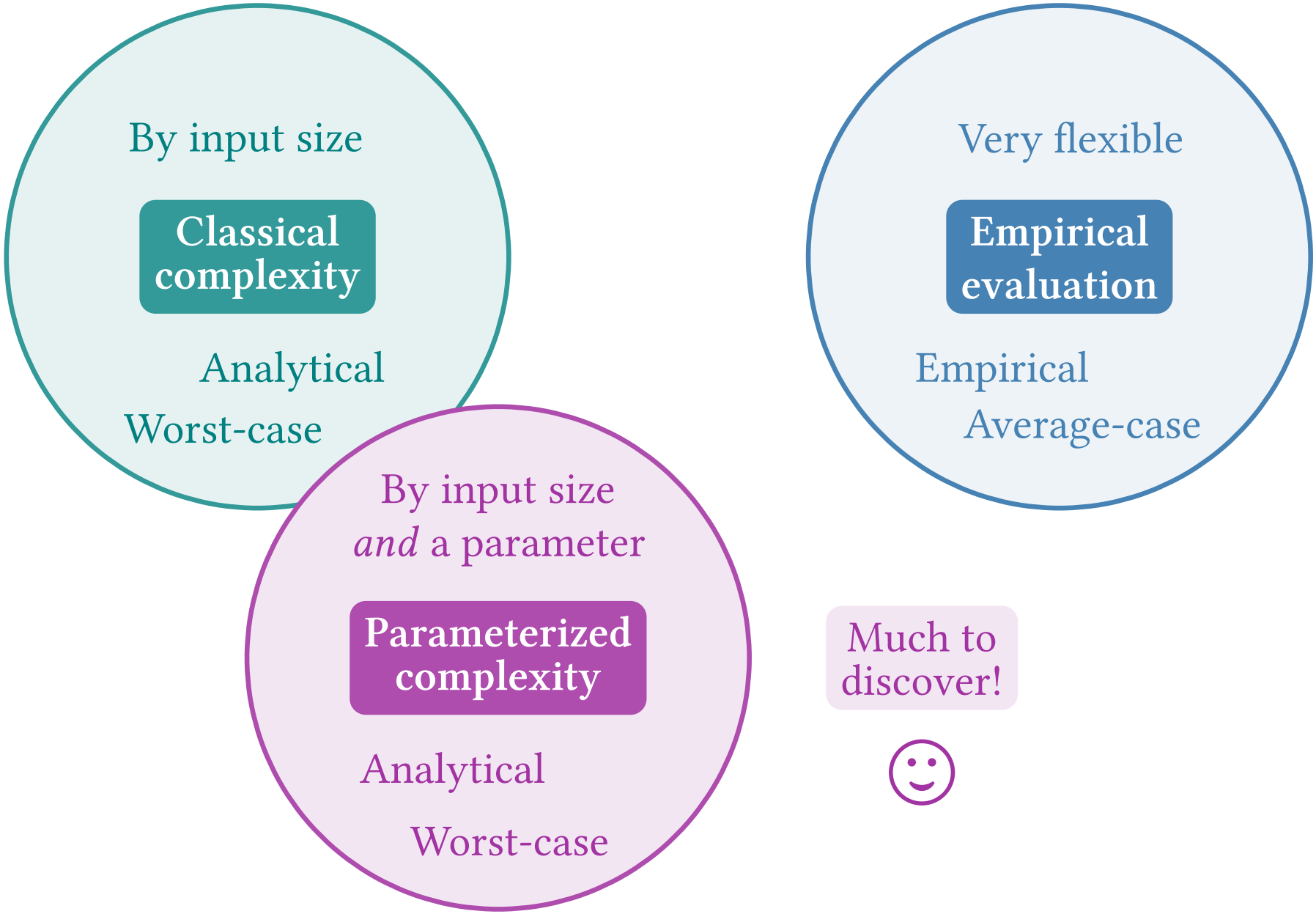
Give us the take-home message already…



Give us the take-home message already…



Give us the take-home message already…



$\forall$Thank you!

$\exists$Questions?