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 is \textit{fixed-parameter analysis} (or parameterized complexity)?
What is \textit{fixed-parameter analysis} (or parameterized complexity)?

\begin{quote}
Complexity as a function of both the \textit{input size} and a problem-specific \textit{parameter}.
\end{quote}
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 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.
Input: A constrained-deadline sporadic task set $\mathcal{T}$.

Question: Is $\mathcal{T}$ FP-schedulable on a single processor?
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!
Input: A constrained-deadline sporadic task set $\mathcal{T}$.

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

This problem is NP-complete!

$\Rightarrow$

If $P \neq NP$, 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

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

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

This problem is NP-complete!

$\Rightarrow$

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

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

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

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

Parameterization 2

$k_2 = \text{number of tasks in } \mathcal{T}$
**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**
  
  $k_1 = \text{max numerical value in } \mathcal{T}$

- **Parameterization 2**
  
  $k_2 = \text{number of tasks in } \mathcal{T}$

Is FP-schedulability “tractable” when $k_1$ or $k_2$ are small?
Fixed-Parameter Tractable (FPT)

FPT is the class of “tractable” parameterized problems.
FPT is the class of “tractable” parameterized problems.

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

- $n$ — size of the input
- $k$ — parameter
- $f$ — computable function
FPT is the class of “tractable” parameterized problems.

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

- $n$ — 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 \textit{fpt-algorithm} runs in time $O(f(k) \times \text{poly}(n))$.

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

Parameter isolated in its own factor

Otherwise polynomial in input size
FPT is the class of “tractable” parameterized problems.

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

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

Parameter isolated in its own factor

Otherwise polynomial in input size
A familiar decision problem

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

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

$k_1 = \text{max numerical value in } \mathcal{T}$ \hspace{1cm} $k_2 = \text{number of tasks in } \mathcal{T}$
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?

$k_1 = \max \text{ numerical value in } \mathcal{T}$ \hspace{1cm} $k_2 = \text{ number of tasks in } \mathcal{T}$

In FPT? \hspace{1cm} $O(f(k_1) \times \text{poly}(n))$?
Parameterizing the FP-schedulability problem

A familiar decision problem

Input: A constrained-deadline sporadic task set $T$.

Question: Is $T$ FP-schedulable on a single processor?

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

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

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

Yes! $O(k_1 \times n^2)$ using RTA
**Parameterizing the FP-schedulability problem**

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

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

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

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

Yes!

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

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

In FPT? $O(f(k_2) \times \text{poly}(n))$?
**Response-Time Analysis (RTA)**

<table>
<thead>
<tr>
<th></th>
<th>$C_i$</th>
<th>$D_i$</th>
<th>$T_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_1$</td>
<td>$x - 1$</td>
<td>$x$</td>
<td>$x$</td>
</tr>
<tr>
<td>$\tau_2$</td>
<td>$x$</td>
<td>$x^2$</td>
<td>$x^2$</td>
</tr>
</tbody>
</table>
### Response-Time Analysis (RTA)

<table>
<thead>
<tr>
<th></th>
<th>$C_i$</th>
<th>$D_i$</th>
<th>$T_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_1$</td>
<td>$x - 1$</td>
<td>$x$</td>
<td>$x$</td>
</tr>
<tr>
<td>$\tau_2$</td>
<td>$x$</td>
<td>$x^2$</td>
<td>$x^2$</td>
</tr>
</tbody>
</table>

$$R_i^{(k+1)} = C_i + \sum_{j \in \text{hp}(i)} \left\lceil \frac{R_i^{(k)}}{T_j} \right\rceil \times C_j$$
**Response-Time Analysis (RTA)**

<table>
<thead>
<tr>
<th></th>
<th>$C_i$</th>
<th>$D_i$</th>
<th>$T_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_1$</td>
<td>$x - 1$</td>
<td>$x$</td>
<td>$x$</td>
</tr>
<tr>
<td>$\tau_2$</td>
<td>$x$</td>
<td>$x^2$</td>
<td>$x^2$</td>
</tr>
</tbody>
</table>

$$R_i^{(k+1)} = C_i + \sum_{j \in hp(i)} \left\lfloor \frac{R_i^{(k)}}{T_j} \right\rfloor \times C_j$$
Response-Time Analysis (RTA)

<table>
<thead>
<tr>
<th></th>
<th>( C_i )</th>
<th>( D_i )</th>
<th>( T_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \tau_1 )</td>
<td>( x - 1 )</td>
<td>( x )</td>
<td>( x )</td>
</tr>
<tr>
<td>( \tau_2 )</td>
<td>( x )</td>
<td>( x^2 )</td>
<td>( x^2 )</td>
</tr>
</tbody>
</table>

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

\[
\begin{align*}
R_2^{(0)} &= 2x - 1 \\
R_2^{(1)} &= 3x - 2 \\
R_2^{(2)} &= 4x - 3 \\
\vdots \\
R_2^{(x-1)} &= x^2
\end{align*}
\]
Response-Time Analysis (RTA)

<table>
<thead>
<tr>
<th></th>
<th>(C_i)</th>
<th>(D_i)</th>
<th>(T_i)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\tau_1)</td>
<td>(x - 1)</td>
<td>(x)</td>
<td>(x)</td>
</tr>
<tr>
<td>(\tau_2)</td>
<td>(x)</td>
<td>(x^2)</td>
<td>(x^2)</td>
</tr>
</tbody>
</table>

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

\[
\begin{align*}
R_2^{(0)} &= 2x - 1 \\
R_2^{(1)} &= 3x - 2 \\
R_2^{(2)} &= 4x - 3 \\
& \vdots \\
R_2^{(x-1)} &= x^2
\end{align*}
\]
**Response-Time Analysis (RTA)**

<table>
<thead>
<tr>
<th></th>
<th>$C_i$</th>
<th>$D_i$</th>
<th>$T_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_1$</td>
<td>$x - 1$</td>
<td>$x$</td>
<td>$x$</td>
</tr>
<tr>
<td>$\tau_2$</td>
<td>$x$</td>
<td>$x^2$</td>
<td>$x^2$</td>
</tr>
</tbody>
</table>

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

- $R_2^{(0)} = 2x - 1$
- $R_2^{(1)} = 3x - 2$
- $R_2^{(2)} = 4x - 3$
- $\vdots$
- $R_2^{(x-1)} = x^2$

$x$ iterations!

RTA is *not* $O(f(\#tasks) \times \text{poly}(n))$ and is therefore *not* an fpt-algorithm here.
Parameterizing the FP-schedulability problem

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

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

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

<table>
<thead>
<tr>
<th>$k_1$</th>
<th>In FPT?</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O(f(k_1) \times \text{poly}(n))$?</td>
<td>Yes! $O(k_1 \times n^2)$ using RTA</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>$k_2$</th>
<th>In FPT?</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O(f(k_2) \times \text{poly}(n))$?</td>
<td></td>
</tr>
</tbody>
</table>
A familiar decision problem

**Input:** A constrained-deadline sporadic task set $T$.

**Question:** Is $T$ FP-schedulable on a single processor?

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

- In FPT? $O(f(k_1) \times \text{poly}(n))?$
  - Yes!
  - $O(k_1 \times n^2)$ using RTA

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

- In FPT? $O(f(k_2) \times \text{poly}(n))?$
  - Not with RTA!
Hyperplanes Exact Test (HET)

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

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

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.

$\implies$

HET runs in $O(f(\#tasks) \times \text{poly}(n))$ time!
**Parameterizing the FP-schedulability problem**

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

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

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

- $k_1$ in FPT?
  - $O(f(k_1) \times \text{poly}(n))$?
  - Yes!
  - $O(k_1 \times n^2)$ using RTA

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

- $k_2$ in FPT?
  - $O(f(k_2) \times \text{poly}(n))$?
  - Yes!
  - $O(2^{k_2} \times n^2)$ using HET

- Not with RTA!
Parameterizing the FP-schedulability 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 $\mathcal{T}$

- $k_1$ = max numerical value in $\mathcal{T}$
- In FPT?
- $O\left( f(k_1) \times \text{poly}(n) \right)$?
- Yes!
- $O(k_1 \times n^2)$ using RTA

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

- $k_2$ = number of tasks in $\mathcal{T}$
- In FPT?
- $O\left( f(k_2) \times \text{poly}(n) \right)$?
- Yes!
- $O(2^{k_2} \times n^2)$ 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?

\[ k_1 = \max \text{numerical value in } \mathcal{T} \]

\[ \begin{align*}
    \text{In FPT?} & \quad \text{Yes!} \\
      O(f(k_1) \times \text{poly}(n)) & \quad O(k_1 \times n^2) \text{ using RTA}
\end{align*} \]

\[ k_2 = \text{number of tasks in } \mathcal{T} \]

\[ \begin{align*}
    \text{In FPT?} & \quad \text{Yes!} \\
      O(f(k_2) \times \text{poly}(n)) & \quad O(2^{k_2} \times n^2) \text{ using HET}
\end{align*} \]

HET is an fpt-alg. even with \( k = \text{number of distinct periods} \)
**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?

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

- **In FPT?** $O\left( f(k_1) \times \text{poly}(n) \right)$?
  - **Yes!**
  - $O(k_1 \times n^2)$ using RTA
- **RTA is an fpt-alg. even with** $k = T_{\text{max}}/T_{\text{min}}$

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

- **In FPT?** $O\left( f(k_2) \times \text{poly}(n) \right)$?
  - **Yes!**
  - $O(2^{k_2} \times n^2)$ using HET
- **HET is an fpt-alg. even with** $k = \text{number of distinct periods}$
Another familiar decision problem

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

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

\[ k = \frac{T_{\text{max}}}{T_{\text{min}}} \]

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!

$k = \frac{T_{\text{max}}}{T_{\text{min}}}$

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

$k = \text{number of tasks}$

Neither PDA nor QPA are fpt-algorithms!
EDF?

Another familiar decision problem

**Input:** A constrained-deadline sporadic task set \( \mathcal{T} \).

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

This problem is \textbf{coNP}-complete!

\[
k = \frac{T_{\text{max}}}{T_{\text{min}}}
\]

Processor Demand Analysis (PDA) is an \textit{fpt}-algorithm for \textit{bounded-utilization} task sets

\[
k = \text{number of tasks}
\]

Neither PDA nor QPA are \textit{fpt}-algorithms!

We can make “small” ILPs that give an \textit{fpt}-algorithm, even for asynchronous tasks
To FPT, or not to FPT

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

\[ \text{FPT} \subseteq \text{para-(co)NP} \]
To FPT, or not to FPT

Can we show that some problems are \textit{not} in FPT?

\[ \text{FPT} \subseteq \text{para-(co)NP} \]

Strict if \( P \neq \text{NP} \)
To FPT, or not to FPT

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

\[ \text{FPT} \subseteq \text{para-(co)NP} \]

Strict if \( P \neq \text{NP} \)

\[ \text{FPT} \subseteq \text{W}[1] \subseteq \text{W}[2] \subseteq \cdots \subseteq \text{XP} \]
To FPT, or not to FPT

Can we show that some problems are \textit{not} in FPT?

Strict if the ETH is true

\[ FPT \subseteq \text{para-(co)NP} \]

Strict if \( P \neq NP \)

\[ FPT \subseteq W[1] \subseteq W[2] \subseteq \cdots \subseteq XP \]
Can we show that some problems are \textit{not} in FPT?

Strict if \( P \neq NP \)

\[
\text{FPT} \subseteq \text{para-}(\text{co})\text{NP}
\]

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

\textbf{Exponential Time Hypothesis (ETH)}

\[ \approx \]

3-SAT cannot be solved in sub-exponential time
Some hardness results

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

Setting

Schedulability

para-coNP-hard w.
• #distinct deadlines
• #distinct WCETs
• max WCET

Schedulability

W[1]-hard w.
• max deadline

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

Setting

Schedulability

para-coNP-hard w.
• #distinct WCETs
• max WCET

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

(Unless P = NP)
**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

This is not para-coNP-hard with #distinct periods! (Unless P = NP)
Some hardness results

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

This is not para-coNP-hard with #distinct periods! (Unless $P=NP$)
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

- 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 \textit{fpt-algorithm} runs in time $O(f(k) \times \text{poly}(n))$.

$n$ — size of the input

$k$ — parameter

$f$ — computable function
Lower bounds on $f$

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

- $n$ — size of the input
- $k$ — parameter
- $f$ — computable function
Lower bounds on $f$

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

$n$ — 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

$$O(2^{o(#tasks)} \times \text{poly}(n)).$$
Lower bounds on $f$

If the ETH holds, then:

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

$$O(2^{o(#	ext{tasks})} \times \text{poly}(n)).$$

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

$$O(2^{o(#	ext{periods})} \times \text{poly}(n)).$$
Give us the take-home message already…
Give us the take-home message already…

Classical complexity

Empirical evaluation
Give us the take-home message already...

Classical complexity
- Analytical
  - Worst-case

Empirical evaluation

Much to discover!
Give us the take-home message already...

- Classical complexity
  - Analytical
    - Worst-case

- By input size

- Empirical evaluation

Much to discover!
Give us the take-home message already…

By input size

Classical complexity
Analytical
Worst-case

Empirical evaluation
Empirical
Average-case

Much to discover!
Give us the take-home message already...

By input size
- Classical complexity
  - Analytical
  - Worst-case

Very flexible
- Empirical evaluation
  - Empirical
  - Average-case

Much to discover!
Give us the take-home message already…

- Classical complexity
  - Analytical
    - Worst-case
  - Parameterized complexity
- Empirical evaluation
  - Empirical
    - Average-case
- Parameterized complexity
  - By input size

Much to discover!
Give us the take-home message already...

Classical complexity
By input size
Analytical
Worst-case

Parameterized complexity
Analytical
Worst-case

Empirical evaluation
Very flexible
Empirical
Average-case
Give us the take-home message already...

- Classical complexity
  - Analytical
  - Worst-case

- Empirical evaluation
  - Empirical
  - Average-case

- Parameterized complexity
  - Analytical
  - Worst-case

By input size

Very flexible

Much to discover!
Give us the take-home message already...

- **Classical complexity**
  - Analytical
  - Worst-case

- **Empirical evaluation**
  - Empirical
  - Average-case

- **Parameterized complexity**
  - Analytical
  - Worst-case

- **Parameterized complexity**
  - Analytical
  - Worst-case

By input size

Very flexible

and a parameter
Give us the take-home message already...

By input size
Classical complexity
Analytical Worst-case

By input size and a parameter
Parameterized complexity
Analytical Worst-case

Very flexible
Empirical evaluation
Empirical Average-case

Much to discover! 😊
∀Thank you!

∃Questions?