Learning Small Decision Trees

Konrad Dabrowski, Eduard Eiben, Sebastian Ordyniak,
Giacomo Paesani, and Stefan Szeider

UNIVERSITY OF LEEDS

PCCR Workshop, 31st July 2022

Motivation

Decision Trees:

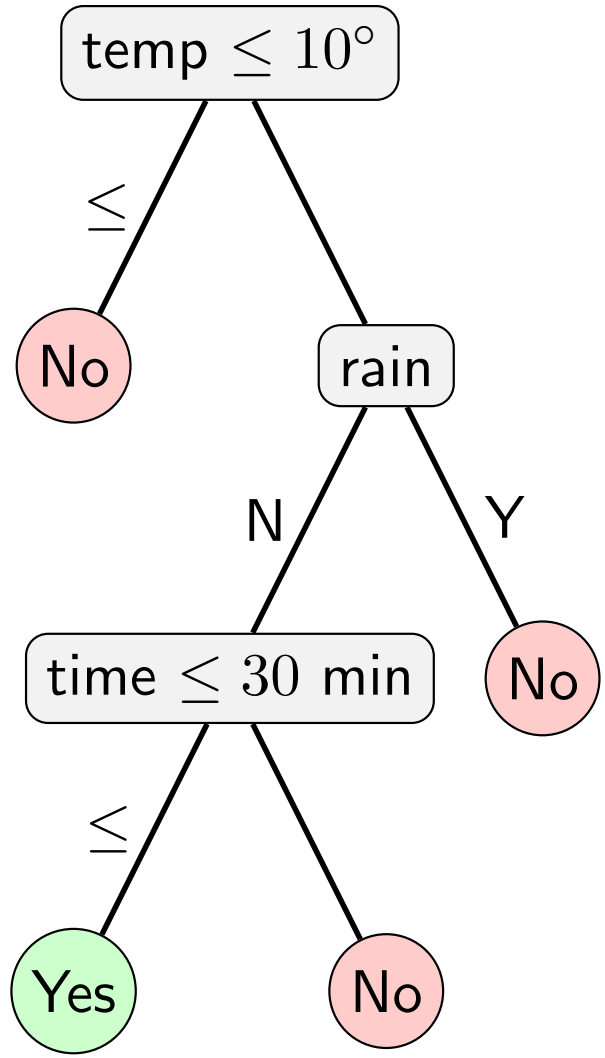
- Useful and prominent tool for the description, classification and generalization of data.
- Due to their simplicity, they provide interpretable models (an aspect becoming increasingly important especially within the context of **explainable AI**).
- Usually **small trees** are preferred (small size or depth) as those provide easier to interpret models and require fewer tests for classification.
- Small trees are also prefered in view of the parsimony principle (Occam's Razor) since expected to generalize better to new data.

Decision Tree: Example

Do I walk to work?

features:

- temperature in °: **temp**,
- does it rain (Y/N): **rain**?
- time to arrival in min: **time**.



Decision Tree: Example

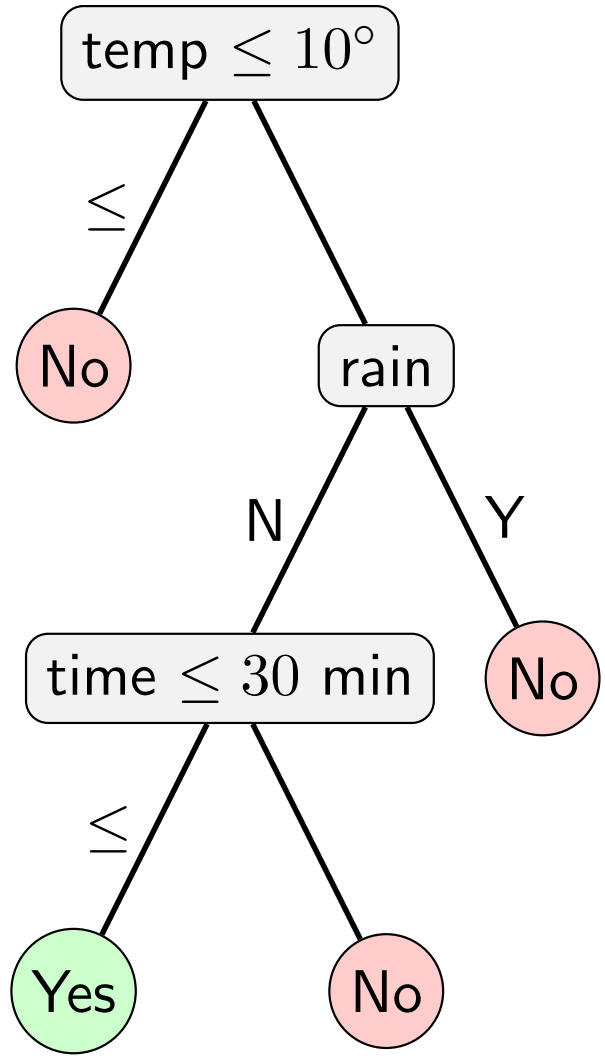
Do I walk to work?

features:

- temperature in °: **temp**,
- does it rain (Y/N): **rain**?
- time to arrival in min: **time**.

Examples:

- temp=$15°$, rain=Y, time=40min?



Decision Tree: Example

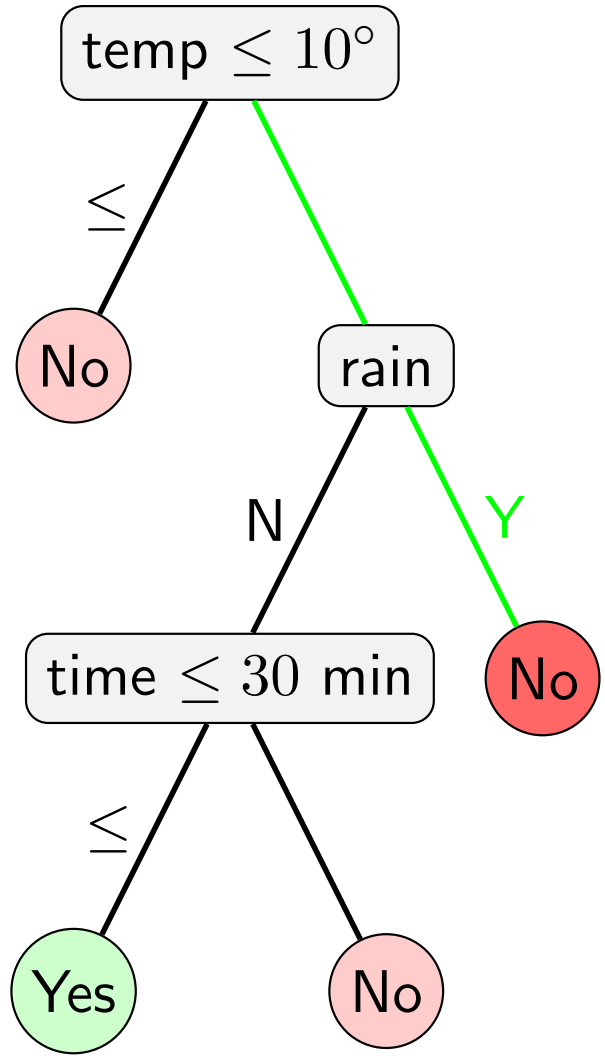
Do I walk to work?

features:

- temperature in °: **temp**,
- does it rain (Y/N): **rain**?
- time to arrival in min: **time**.

Examples:

- temp=$15°$, rain=Y, time=40min? $\rightarrow$ **No**



Decision Tree: Example

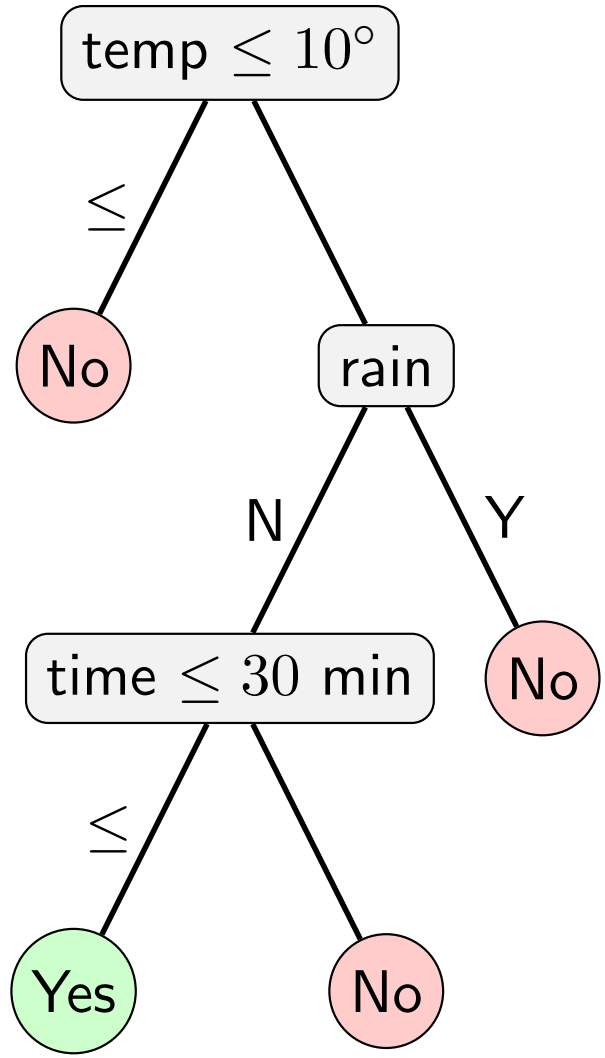
Do I walk to work?

features:

- temperature in °: **temp**,
- does it rain (Y/N): **rain**?
- time to arrival in min: **time**.

Examples:

- temp=$15°$, rain=Y, time=40min? $\rightarrow$ **No**
- temp=$15°$, rain=N, time=30min?



Decision Tree: Example

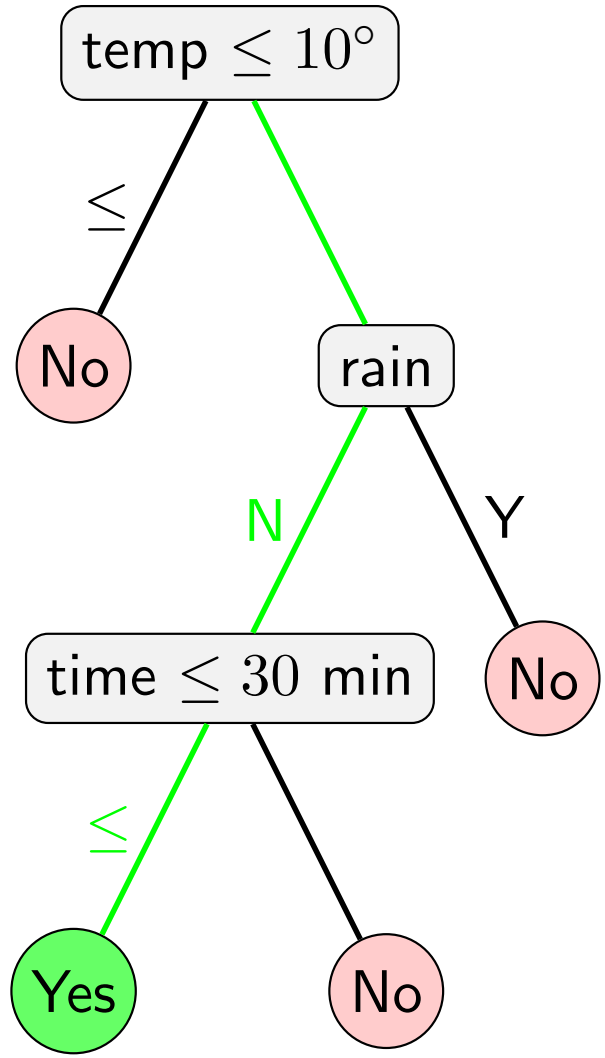
Do I walk to work?

features:

- temperature in °: **temp**,
- does it rain (Y/N): **rain**?
- time to arrival in min: **time**.

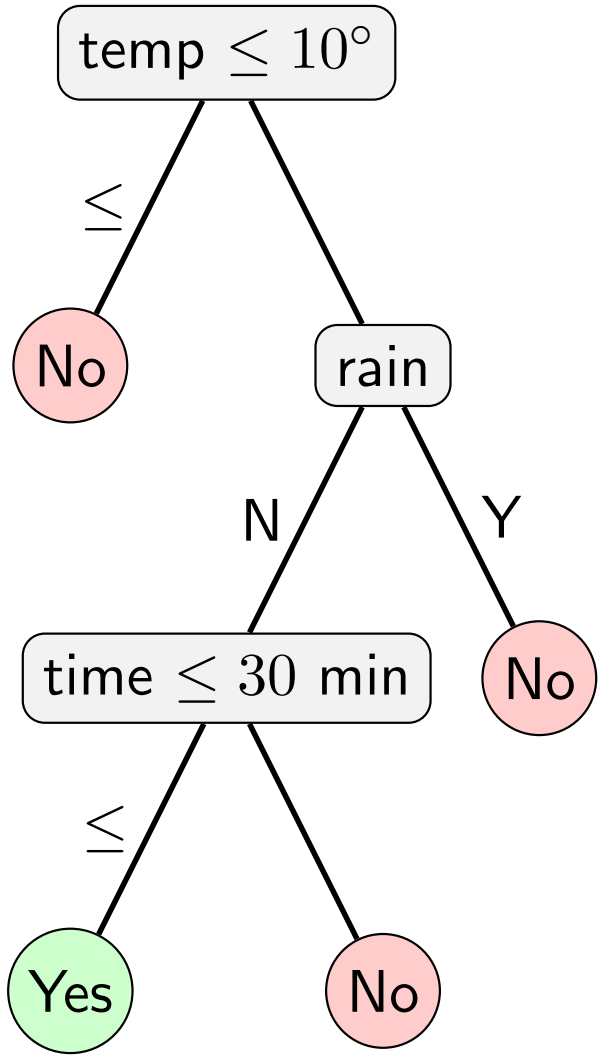
Examples:

- temp=$15°$, rain=Y, time=40min? $\rightarrow$ **No**
- temp=$15°$, rain=N, time=30min? $\rightarrow$ **Yes**



Learning Decision Trees: Example

temp	rain	time	day	walk
3°	Y	10	3	**No**
20°	N	20	2	**Yes**
12°	Y	20	4	**No**
12°	N	15	2	**Yes**
15°	N	45	5	**No**
10°	N	35	3	**No**
21°	N	15	5	**Yes**



A **classification instance (CI)** $\mathbf{E} = \mathbf{E}^- \uplus \mathbf{E}^+$ is the disjoint union of:

- the set $\mathbf{E}^-$ of negative examples and
- the set $\mathbf{E}^+$ of positive examples.

An example $e \in E$ is a function

$$\mathbf{e} : \mathit{feat}(\mathbf{E}) \to \mathit{dom}$$

from the set of features $\mathit{feat}(E)$ to the possibly infinite (but totally ordered) domain.

temp	rain	time	day	walk
$3°$	Y	10	3	E^-
$20°$	N	20	2	E^+
$12°$	Y	20	4	E^-
$12°$	N	15	2	E^+
$15°$	N	45	5	E^-
$10°$	N	35	3	E^-
$21°$	N	15	5	E^+

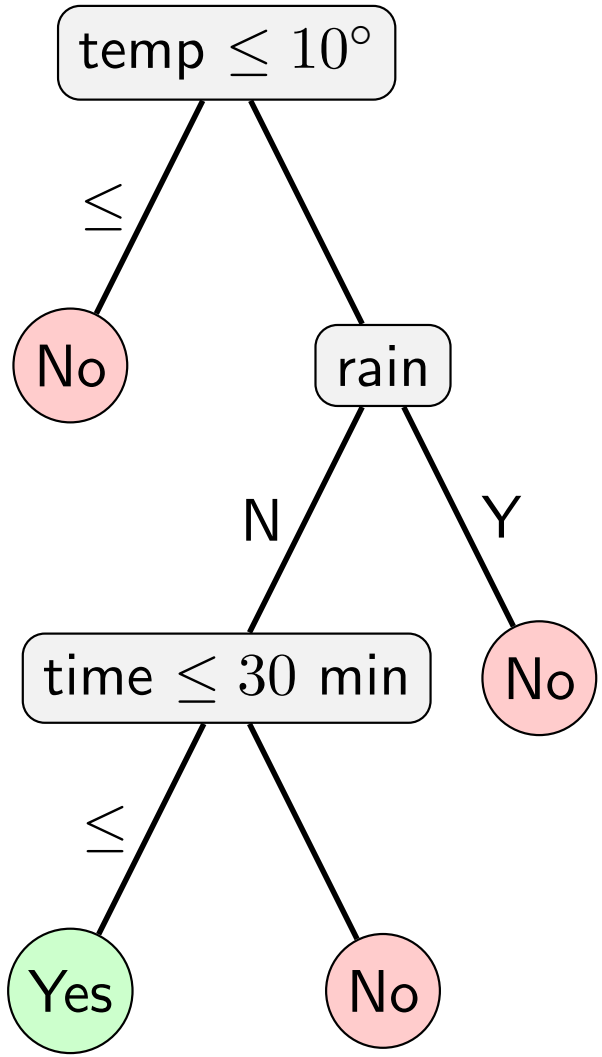
Decision Trees: Setting

We assume that every feature has a possibly infinite but totally ordered domain.

We consider **binary** decision trees, where every inner node t:

- has a **feature** $\mathit{feat}(\mathbf{t})$ and
- a **threshold** $\lambda(\mathbf{t})$,
- the left (right) child of t takes all examples such that $\mathit{feat}(\mathbf{t}) \leq \lambda(\mathbf{t})$ ($\mathit{feat}(\mathbf{t}) > \lambda(\mathbf{t})$).

T is a DT for E if every leaf of T takes a uniform set of examples (i.e., a subset of E^+ or of E^-).



Considered Problems

Finding some DT for a classification instance is trivial:

Simply take a DT that performs all possible tests for all possible features.

However, one is usually interested in finding a **smallest possible DT**, either w.r.t:

- size (i.e., number of nodes) or
- depth/height (i.e., length of a longest root to leaf path)

Considered Problems

Minimum Decision Tree Size

Input: A classification instance E and an integer s.
Question: Does E have a DT of size at most s?

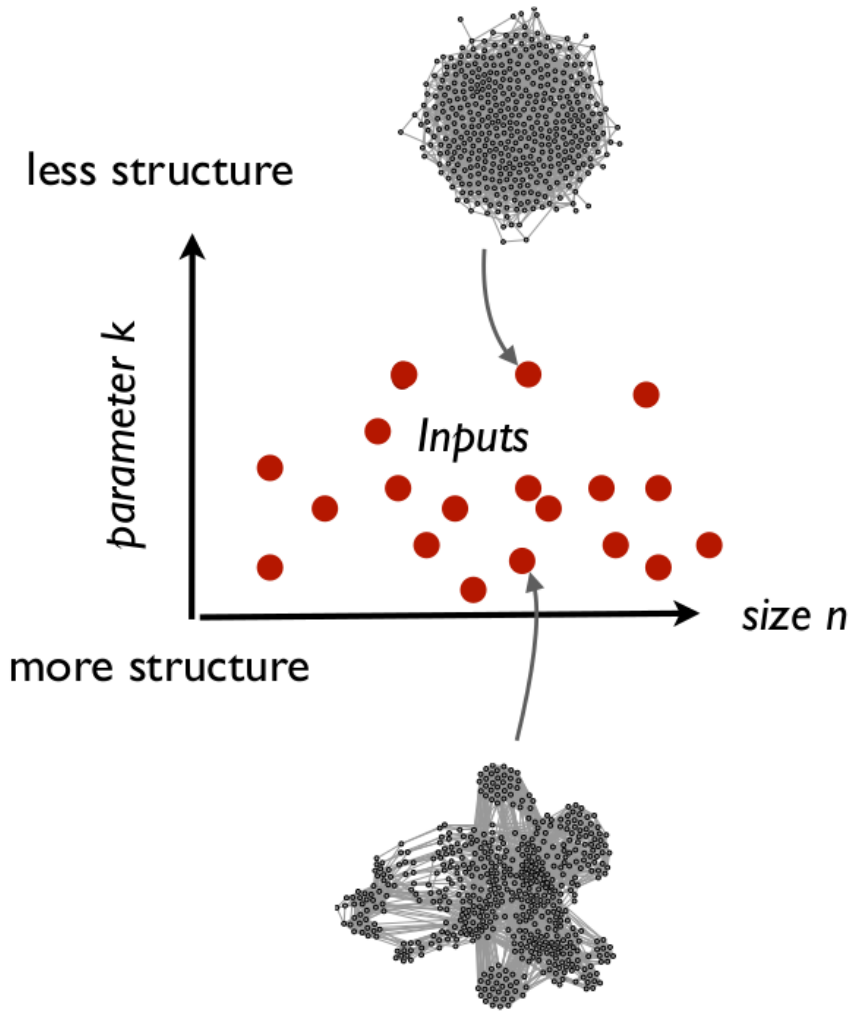
Minimum Decision Tree Depth

Input: A classification instance E and an integer d.
Question: Does E have a DT of depth (height) at most d?

Both problems are well-known to be **NP**-complete and thus far mostly heuristic approaches and approximation algorithms have been considered.

Parameterized Complexity

- measures the complexity of a problem in terms of
  - the **input size** $\mathbf{n}$ and
  - the **parameter** $\mathbf{k}$
- aim is to design exact algorithms for arbitary large instances as long as those instances are highly structured



Parameterized Complexity: Levels of Tractability

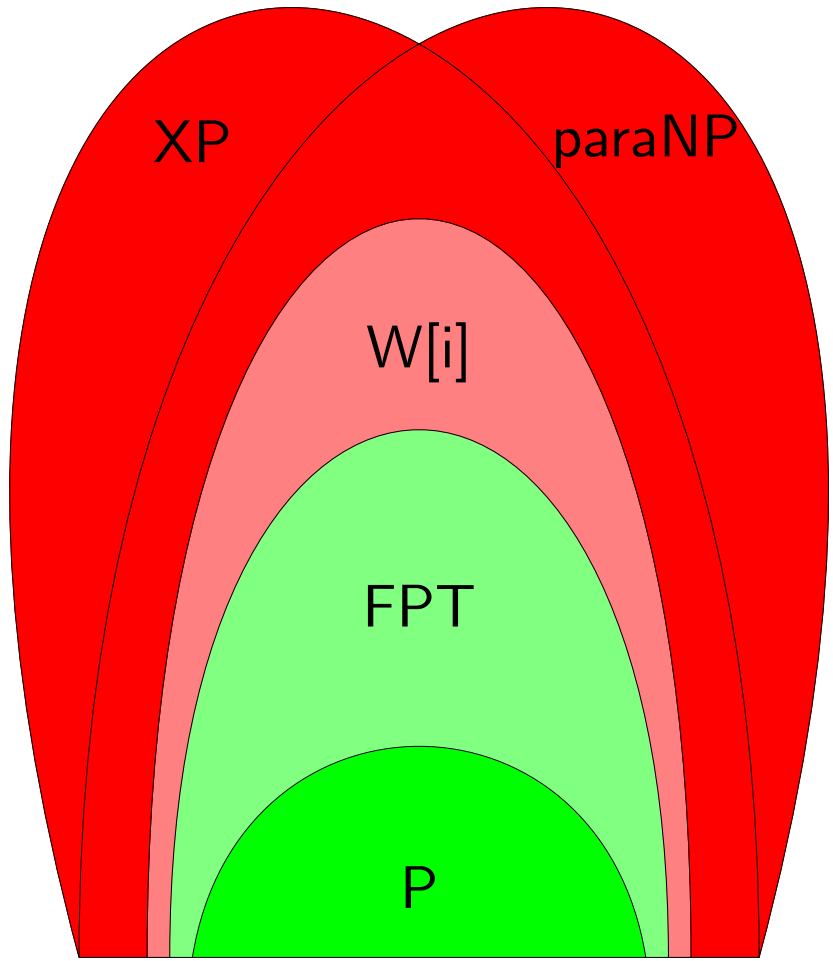
- **fixed-parameter tractable** (**FPT**):

$$\mathbf{f(k)n^{\mathcal{O}(1)}}$$

- **W[i]-hard** $\approx$ unlikely to be **FPT**,
- **XP-tractable** (**XP**) $\approx$ polynomial-time tractable for constant k:

$$\mathbf{n^{f(k)}}$$

- **para-NP-hard** $\approx$ **NP-hard** for constant parameter value.



Considered Parameters

- solution size (*sol*), i.e., either size or depth,

Considered Parameters

- solution size (*sol*), i.e., either size or depth,
- maximum number of domain values of any feature ($\mathbf{D}_{\max}$),

Considered Parameters

- solution size (*sol*), i.e., either size or depth,
- maximum number of domain values of any feature ($\mathbf{D}_{\max}$),
- the maximum hamming distance between any positive and negative example ($\delta_{\max}$), i.e., the maximum number of features in which any positive and negative example differ:

Considered Parameters

- solution size (*sol*), i.e., either size or depth,
- maximum number of domain values of any feature ($\mathbf{D}_{\max}$),
- the maximum hamming distance between any positive and negative example ($\delta_{\max}$), i.e., the maximum number of features in which any positive and negative example differ:

$$\max_{e^+ \in E^+ \wedge e^- \in E^-} |\delta(e^+, e^-)|$$

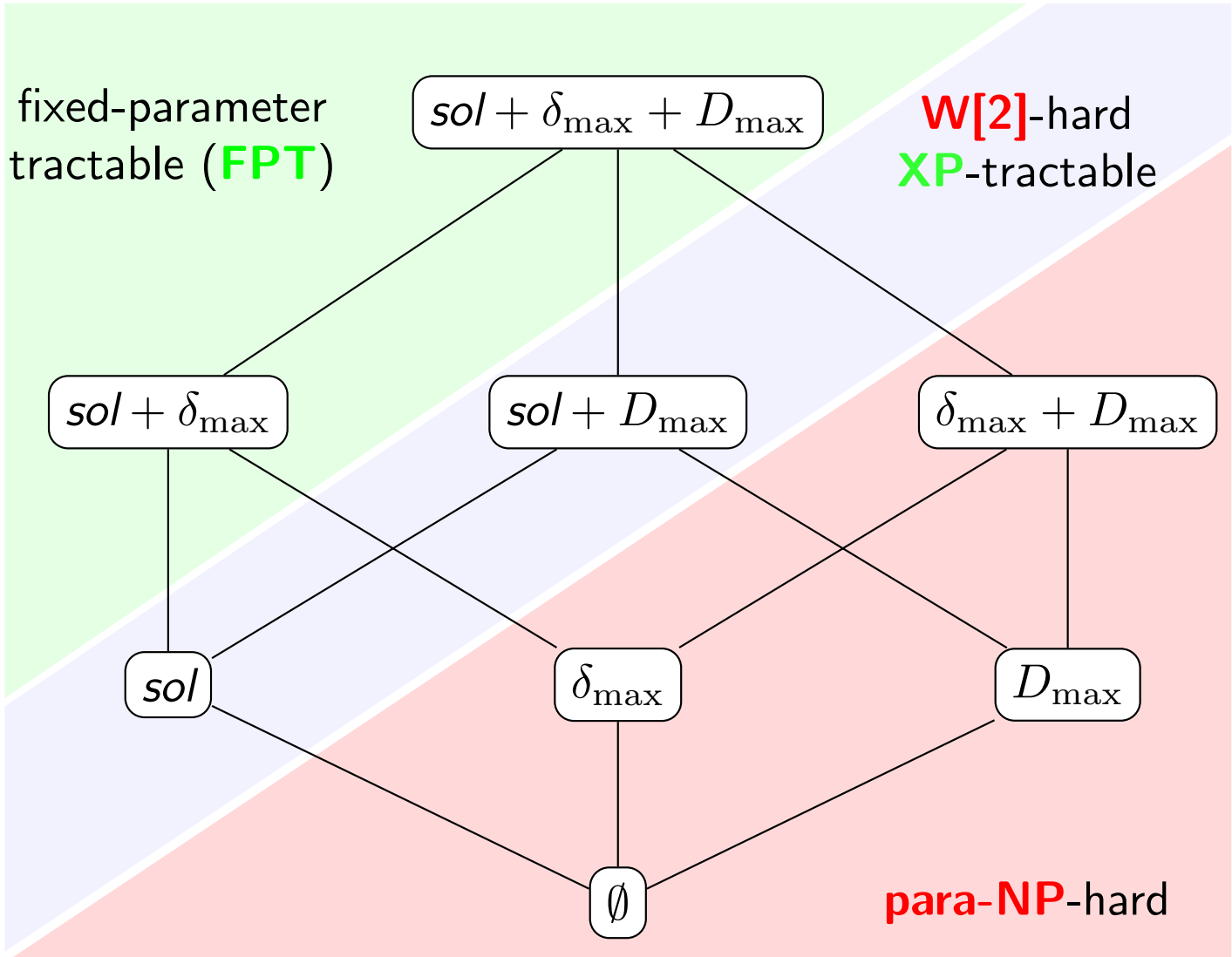
where $\delta(e', e'')$ is the set of features f where $e'(f) \neq e''(f)$.

Practical Relevance of Parameters

Name	$\lvert E\rvert$	$\lvert F\rvert$	$\delta_{\max}$
Append-tis	106	531	14
Australian	690	1164	24
Backache	180	476	31
Car	1728	22	12
Cancer	683	90	18
Colic	368	416	43
Cleve	303	396	23
Haberman	300	93	6

Name	$\lvert E\rvert$	$\lvert F\rvert$	$\delta_{\max}$
Heart-statlog	270	382	23
Hepatitis	155	362	34
HouseVotes	435	17	16
Hungarian	294	331	24
New-thyroid	215	335	10
Promoters	106	335	106
Shuttle	14500	692	18
Spect	250	23	22

Overview of Results



Support Sets

Let $E = E^+ \uplus E^-$ be a CI.

- a feature f **distiniguishes** between two examples e and e' if $e(f) \neq e'(f)$,
- a set S of features is a **support set** for E if for every pair (e^+, e^-) of a positive example $e^+ \in E^+$ and a negative example $e^- \in E^-$ there is a feature in S that distinguishes e^+ and e^-.

Observation

The set of features of every DT for E forms a support set.

Support Set

Input: A CI E over a set of features F and an integer k.
Question: Does F have a support set for E of size at most k?

Support Sets and Hitting Sets

Hitting Set

Input: A family $\mathcal{F}$ of sets over a universe U and an integer k.
Question: Does $\mathcal{F}$ have a hitting set of size at most k?

Remarks

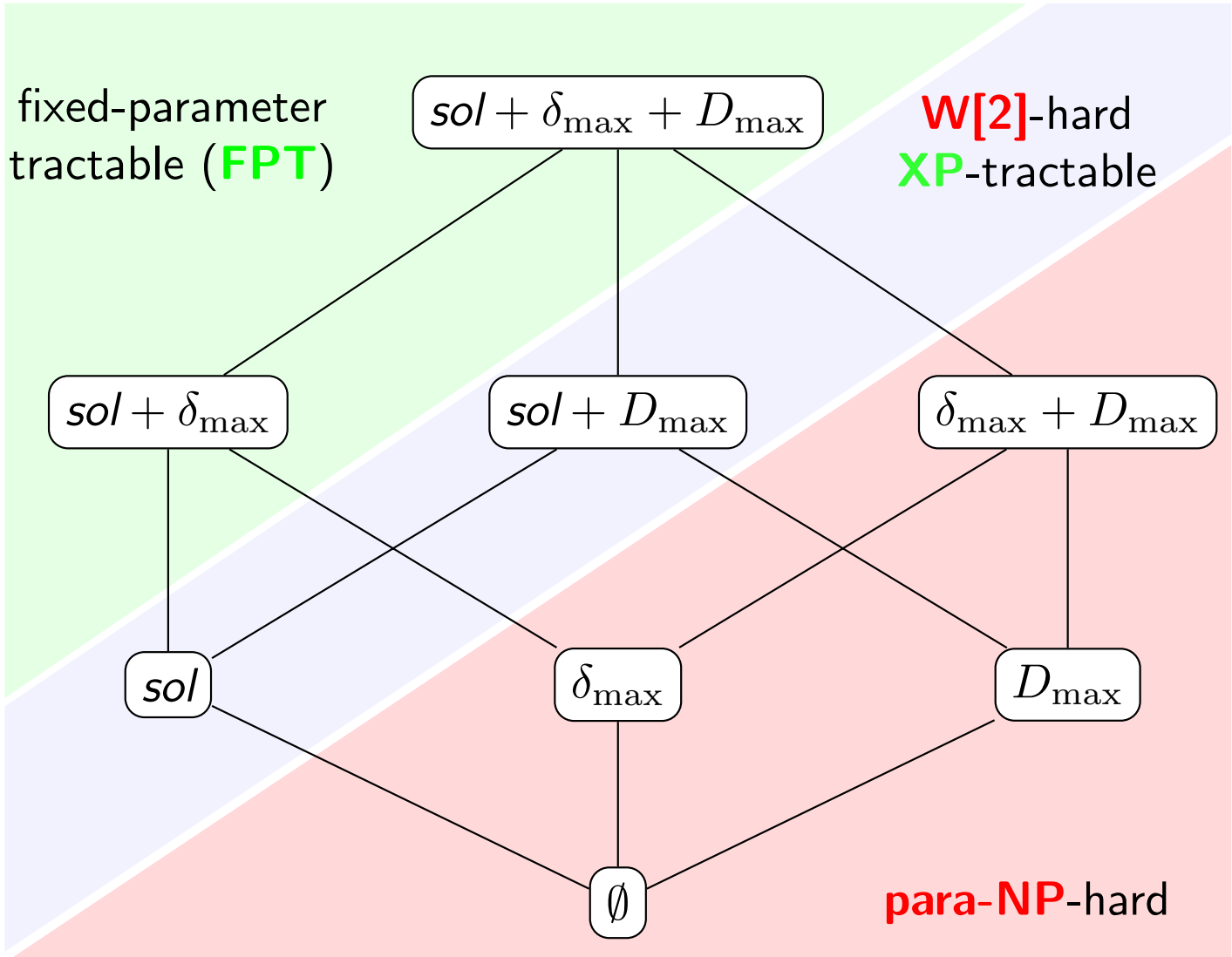
- A **hitting set** for $\mathcal{F}$ is a subset H of U such that $F \cap H \neq \emptyset$ for every $F \in \mathcal{F}$.
- Hitting Set is well-known to be **NP**-complete and **W[2]**-complete parameterized by k.
- Hitting Set and Support Set are equivalent problems.

Support Sets and Hitting Sets

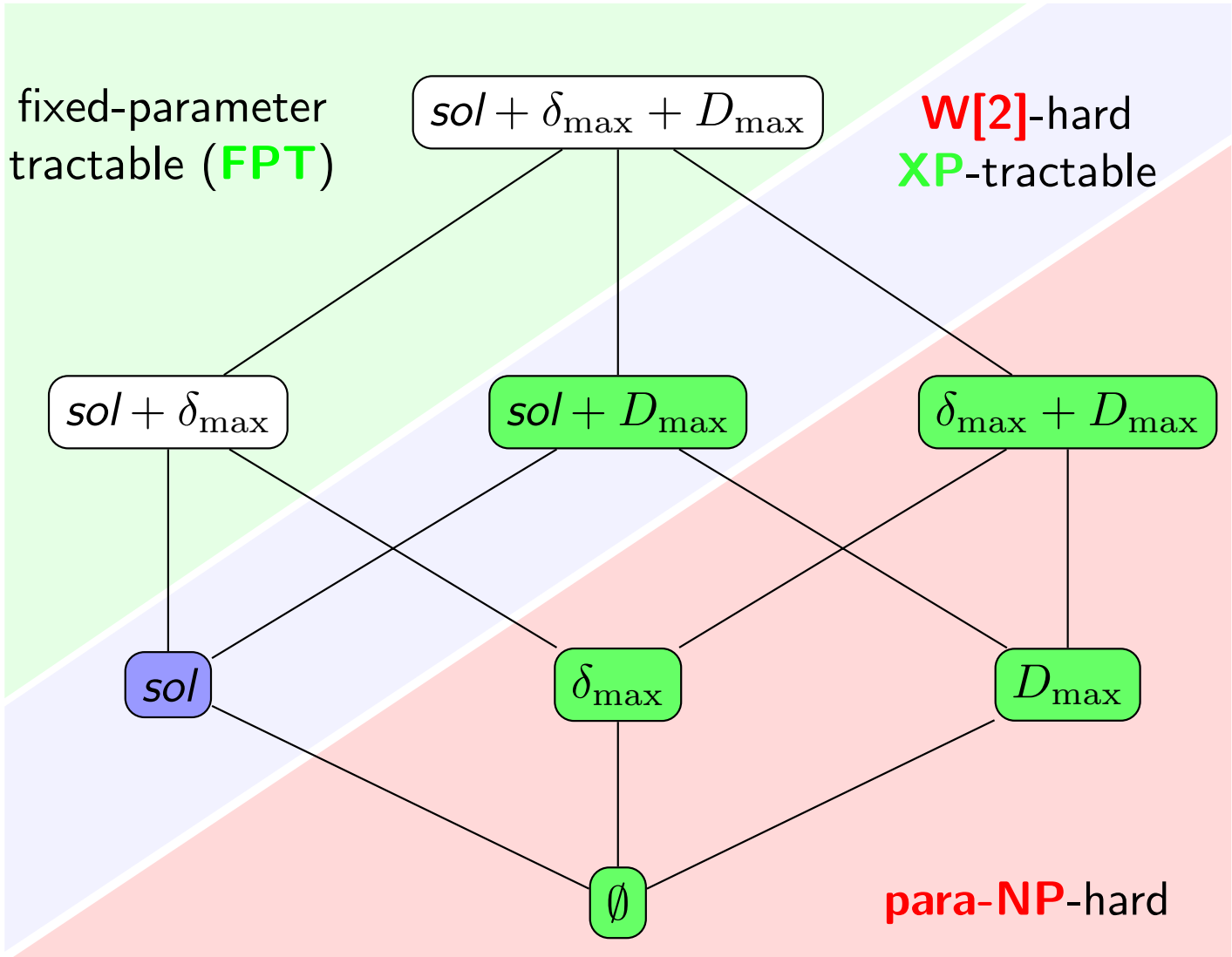
Corollaries

- Minimum Support Set is **FPT** parametrized by $sol + \delta_{\max}$.
- Minimum Support Set is **para-NP**-complete for $\delta_{\max} + D_{\max}$ and **W[2]**-complete parameterized by $sol + D_{\max}$.
- Minimum Decision Tree Size and Depth are **para-NP**-hard for $\delta_{\max} + D_{\max}$ and **W[2]**-hard parameterized by $sol + D_{\max}$.

Overview of Results



Overview of Results



XP-algorithm for *sol*

Algorithm for Minimum Decision Tree Size:

- enumerate all sets P of at most s-many (feat, λ)-pairs that are used in the DT (at most n^{2s} possibilities)
- enumerate all DTs T of size at most s using only the pairs in P (at most s^s possibilities)
- check whether T is a decision tree for E

If any such T is a DT for E, then output T, otherwise output **No**.

Runtime: $\mathcal{O}(n^{2s})$

XP-algorithm for *sol*

Algorithm for Minimum Decision Tree Size:

- enumerate all sets P of at most s-many (feat, λ)-pairs that are used in the DT (at most n^{2s} possibilities)
- enumerate all DTs T of size at most s using only the pairs in P (at most s^s possibilities)
- check whether T is a decision tree for E

If any such T is a DT for E, then output T, otherwise output **No**.

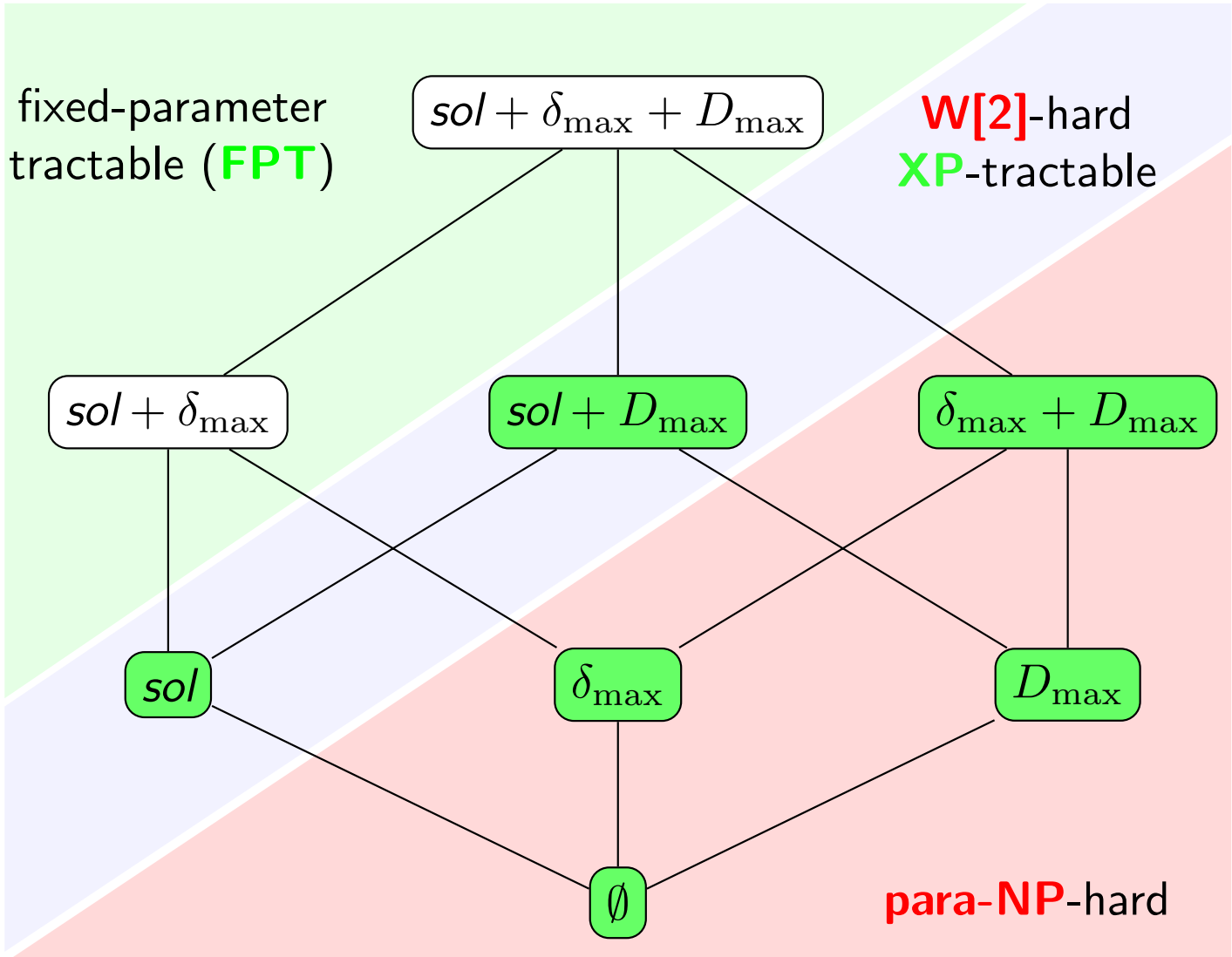
Runtime: $\mathcal{O}(n^{2s})$

Remark

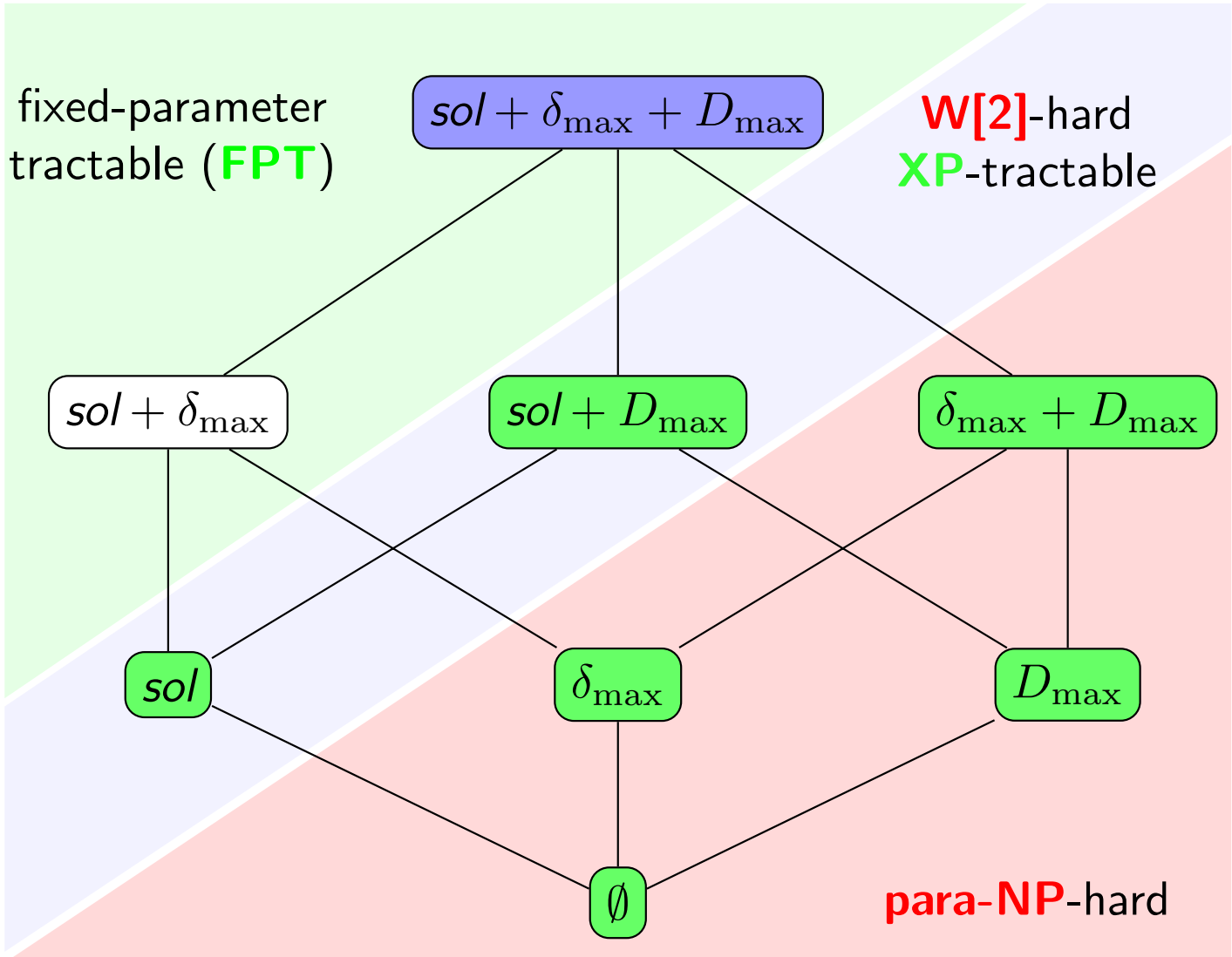
The algorithm for Minimum Decision Tree Depth works essentially the same after observing that:

(*) Every DT of depth d has size at most 2^d.

Overview of Results



Overview of Results



FPT-algorithm for $sol + \delta_{\max} + D_{\max}$

- for simplicity, we will only show the algorithm for Minimum Decision Tree Size
- the algorithm consists of two parts:
  - (P1) **feature selection**, i.e., enumerate all sets S of features that could potentially be used by an optimal DT
  - (P2) **threshold selection**, i.e., given a set S of features find a DT of minimum size using only features in S.

Feature Selection

Recall

The set of features of every DT for E forms a support set.

Moreover, using the equivalence between Hitting Sets and Support Sets, one can obtain that:

Theorem

There is an algorithm that in time $\mathcal{O}(\delta_{\max}^{s}|E|)$ enumerates all of the at most $\delta_{\max}^{s}$ inclusion-wise **minimal support sets** of size at most s for a CI E.

Question

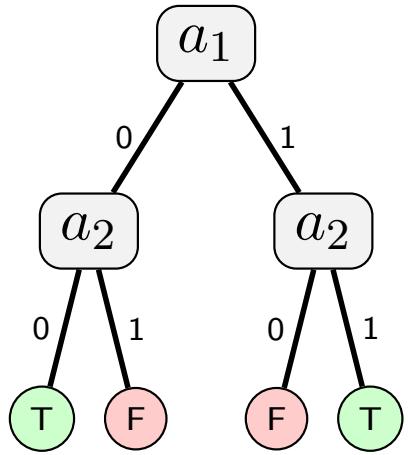
Is it sufficient to only consider **minimal** support sets?

Feature Selection

a_1	a_2	b_1	b_2	x	
0	0	0	0	0	T
0	1	0	0	0	F
1	0	0	0	0	F
1	1	0	0	0	T
0	0	0	0	1	T
0	0	0	1	1	F
0	0	1	0	1	F
0	0	1	1	1	T

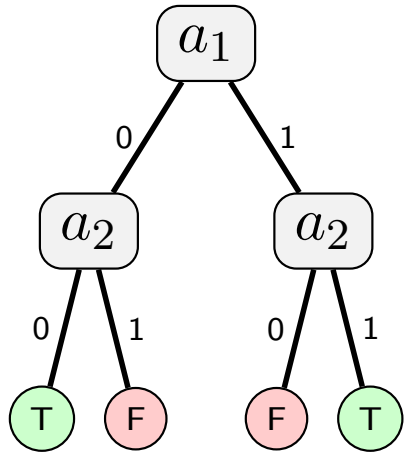
Feature Selection

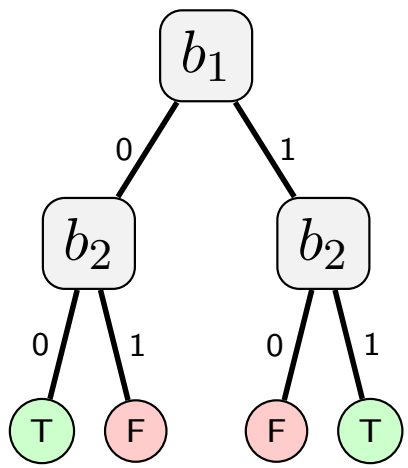
a_1	a_2	b_1	b_2	x	
0	0	0	0	0	T
0	1	0	0	0	F
1	0	0	0	0	F
1	1	0	0	0	T
0	0	0	0	1	T
0	0	0	1	1	F
0	0	1	0	1	F
0	0	1	1	1	T



Feature Selection

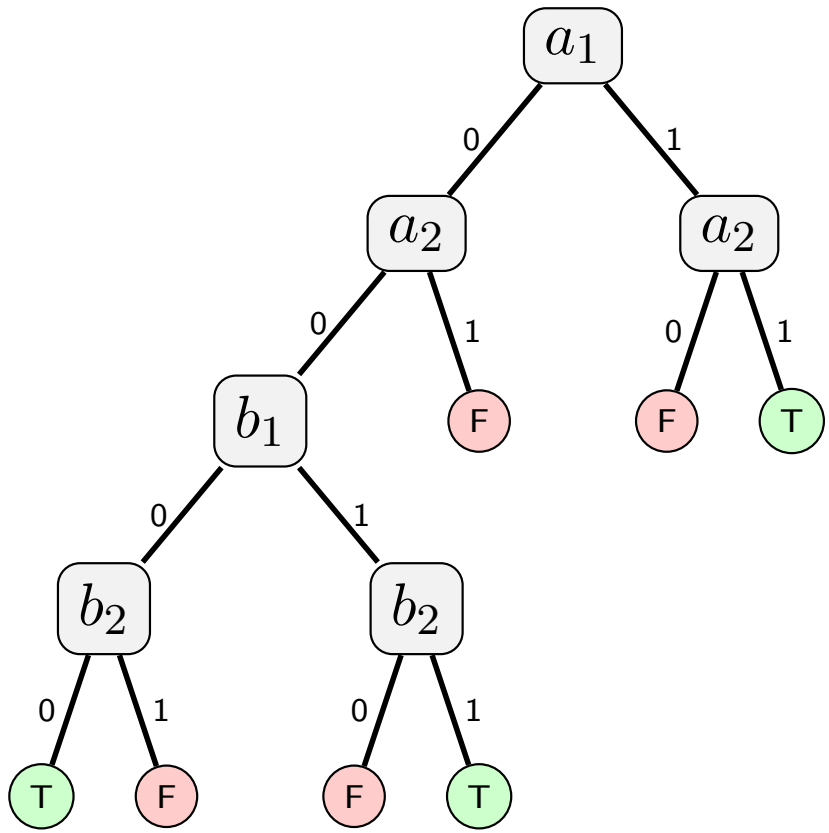
a_1	a_2	b_1	b_2	x	
0	0	0	0	0	T
0	1	0	0	0	F
1	0	0	0	0	F
1	1	0	0	0	T
0	0	0	0	1	T
0	0	0	1	1	F
0	0	1	0	1	F
0	0	1	1	1	T





Feature Selection

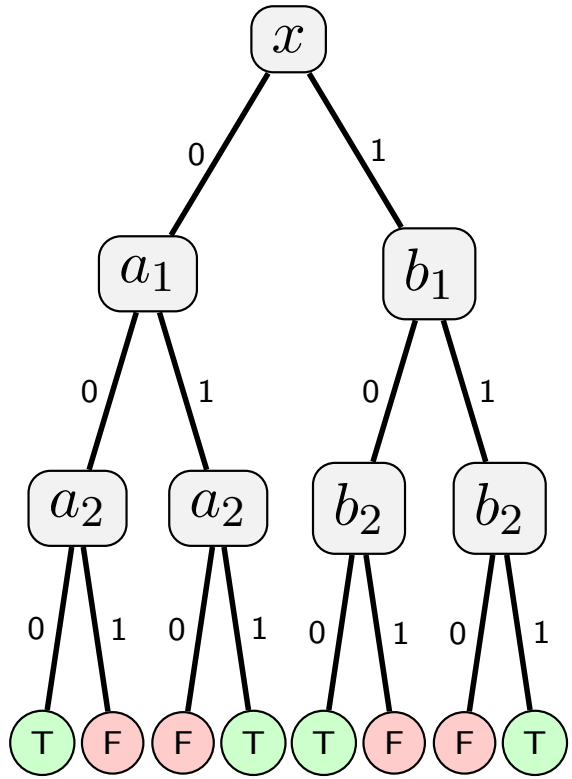
a_1	a_2	b_1	b_2	x	
0	0	0	0	0	T
0	1	0	0	0	F
1	0	0	0	0	F
1	1	0	0	0	T
0	0	0	0	1	T
0	0	0	1	1	F
0	0	1	0	1	F
0	0	1	1	1	T



optimum depth 5 DT using only features a_1, a_2, b_1, b_2.

Feature Selection

a_1	a_2	b_1	b_2	x	
0	0	0	0	0	T
0	1	0	0	0	F
1	0	0	0	0	F
1	1	0	0	0	T
0	0	0	0	1	T
0	0	0	1	1	F
0	0	1	0	1	F
0	0	1	1	1	T



optimum depth 4 DT using all features.

Feature Selection

a_1	a_2	b_1	b_2	x	
0	0	0	0	0	T
0	1	0	0	0	F
1	0	0	0	0	F
1	1	0	0	0	T
0	0	0	0	1	T
0	0	0	1	1	F
0	0	1	0	1	F
0	0	1	1	1	T

- every support set has to contain the features a_1, a_2, b_1, b_2,
- the set $\{a_1, a_2, b_1, b_2\}$ is a support set,
- the set $\{a_1, a_2, b_1, b_2\}$ is the only inclusion-wise minimal support set,
- however, any DT using only $\{a_1, a_2, b_1, b_2\}$ has depth at least 5 and ...
- ... there is a DT of depth 4 (that additionally uses x)

Feature Selection

a_1	a_2	b_1	b_2	x	
0	0	0	0	0	T
0	1	0	0	0	F
1	0	0	0	0	F
1	1	0	0	0	T
0	0	0	0	1	T
0	0	0	1	1	F
0	0	1	0	1	F
0	0	1	1	1	T

- every support set has to contain the features a_1, a_2, b_1, b_2,
- the set $\{a_1, a_2, b_1, b_2\}$ is a support set,
- the set $\{a_1, a_2, b_1, b_2\}$ is the only inclusion-wise minimal support set,
- however, any DT using only $\{a_1, a_2, b_1, b_2\}$ has depth at least 5 and ...
- ... there is a DT of depth 4 (that additionally uses x)

Therefore, considering only inclusion-wise minimal support sets is not sufficient to obtain a DT of minimum depth.

(The same can be shown for a DT of minimum size.)

Feature Selection

- we can still enumerate all inclusion-wise minimal support sets of size at most s,
- as it turns out given a minimal support S, it is possible to enumerate all sets of features U that can be added to S in order to obtain a smaller DT.

Let $S = \{s_1, s_2\}$ be a support set and let $U = \{u_1, u_2\}$.

- every assignments of S gives rise to uniform equivalence classes of examples,
- S cannot distinguish between examples in the same equivalence class.

Definition (Informal)

We say that a set of features U is **useful** for a support set S if it can distinguish between examples in one of the equivalence classes that arise from S.

s_1	s_2	u_1	u_2	
0	0	*	*	
	...			T
0	0	*	*	
0	1	*	*	
	...			F
0	1	*	*	
1	0	*	*	
	...			T
1	0	*	*	
1	1	*	*	
	...			F
1	1	*	*	

Let $S = \{s_1, s_2\}$ be a support set and let $U = \{u_1, u_2\}$.

- every assignments of S gives rise to uniform equivalence classes of examples,
- S cannot distinguish between examples in the same equivalence class.

Definition (Informal)

We say that a set of features U is **useful** for a support set S if it can distinguish between examples in one of the equivalence classes that arise from S.

s_1	s_2	u_1	u_2	
0	0	1	1	
		$\cdots$		T
0	0	0	1	
0	1	0	1	
		$\cdots$		F
0	1	0	0	
1	0	0	0	
		$\cdots$		T
1	0	1	0	
1	1	1	0	
		$\cdots$		F
1	1	1	0	

Useful Sets

Definition (Informal)

We say that a set of features U is **useful** for a support set S if it can distinguish between examples in one of the equivalence classes that arise from S.

Definition (Formal)

We say that U is **useful** for S if for every assignment $\beta : U \to D$, there is an assignment $\alpha : S \to D$ with $E[\alpha] \neq \emptyset$ but $E[\alpha \cup \beta] = \emptyset$.

s_1	s_2	u_1	u_2	
0	0	1	1	
	$\cdots$			T
0	0	0	1	
0	1	0	1	
	$\cdots$			F
0	1	0	0	
1	0	0	0	
	$\cdots$			T
1	0	1	0	
1	1	1	0	
	$\cdots$			F
1	1	1	0	

- $E[\alpha] \approx$ the set of all examples agreeing with α

Finding Useful Sets

Let S be a support set for a CI E.

Definition

A set B of features is a **branching set** for S if $B \cap U \neq \emptyset$ for every useful set U for S.

Definition (Alternative)

A set B of features is a **branching set** for S if for every minimal DT T for E such that $S \subset feat(T)$ then $B \cap (feat(T) \setminus S) \neq \emptyset$.

Lemma

A branching set B for S of size at most $D_{\max}^{|S|} 2\delta_{\max}$ can be computed in polynomial-time.

Feature Selection: The End

We say that a family $\mathcal{F}$ of sets of features is **complete** for E if for every DT T for E of size at most s, there is a set $F \in \mathcal{F}$ such that T uses exactly the variables in F.

Theorem

There is an algorithm that in time $\mathcal{O}((D^s_{\max} 2\delta_{\max})^s |E|)$ computes a complete family of sets of features for E of size at most $(D^s_{\max} 2\delta_{\max})^s$.

Threshold Selection (or Solving CIs with few features)

Threshold Selection

Input: A CI E, an integer s, and a support set S of size at most s.
Question: Is there a DT T of size at most s for E that uses only the features in S?

Remarks

- the problem is clearly **FPT** for $s + D_{\max}$ since we can simply guess the DT T including its features and thresholds
- However, is it also **FPT** for s alone?
- If so, this could be a first step towards resolving the open question whether our problems are **FPT** for $s + \delta_{\max}$ (without $D_{\max}$).

Threshold Selection

- We start by guessing the structure of the DT T including the features used at its inner nodes (this can be achieved in time $\mathcal{O}(s^s)$ since $|S| \leq s$).

- Now, the problem becomes:

  Given a DT T without thresholds, is it possible to assign thresholds to each inner node such that T is a DT for E?

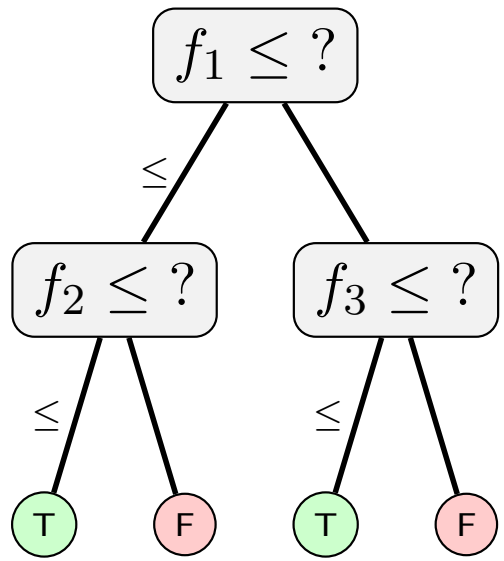
Threshold Selection

Observation

Let r be the root of T with feature f, left child l, and right child r.

Then, increasing the threshold for f:

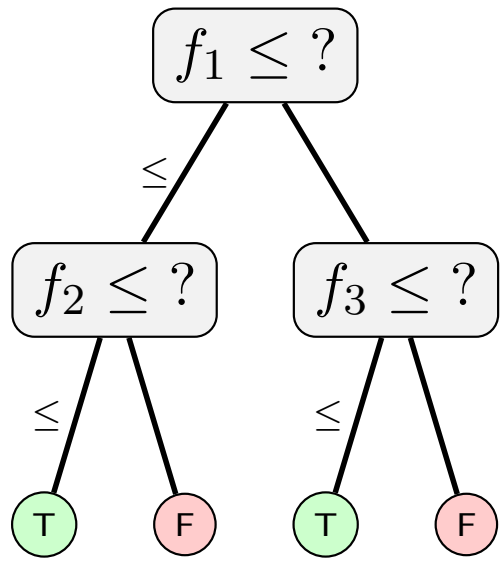
- adds examples that need to be classified by T_l and
- removes examples that need to be classified by T_r.



Threshold Selection

Therefore:

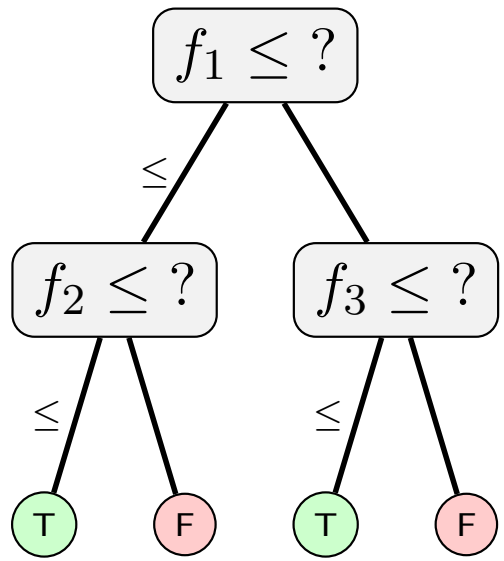
- there is a largest threshold $\lambda_{\max}$ for f such that T_l can be completed into a DT for $E[f \leq \lambda]$ and
- T can be completed into a DT for E if and only if:
  - T_l can be completed into a DT for $E[f \leq \lambda_{\max}]$ and
  - T_r can be completed into a DT for $E[f > \lambda_{\max}]$.



Threshold Selection

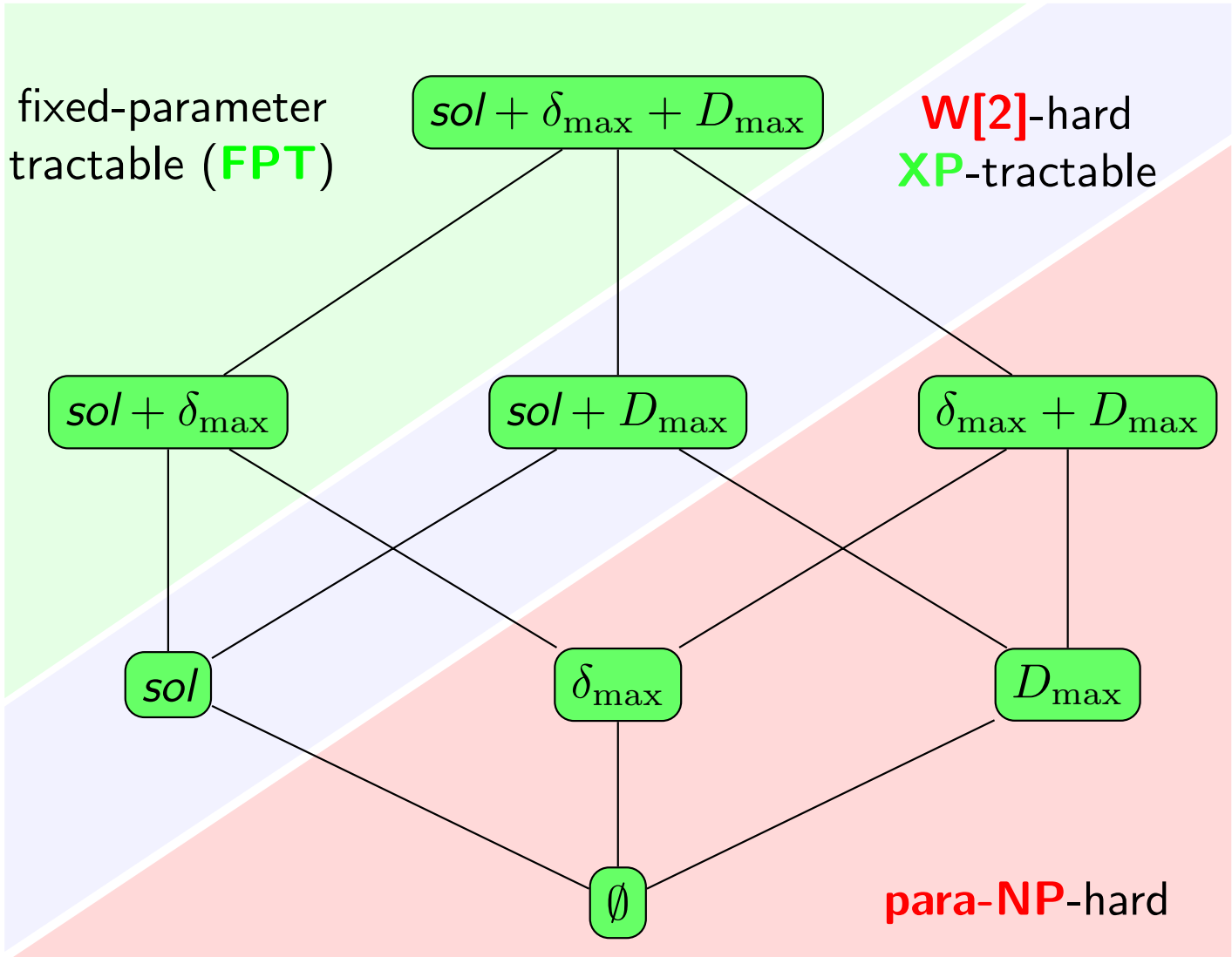
Therefore:

- there is a largest threshold $\lambda_{\max}$ for f such that T_l can be completed into a DT for $E[f \leq \lambda]$ and
- T can be completed into a DT for E if and only if:
  - T_l can be completed into a DT for $E[f \leq \lambda_{\max}]$ and
  - T_r can be completed into a DT for $E[f > \lambda_{\max}]$.

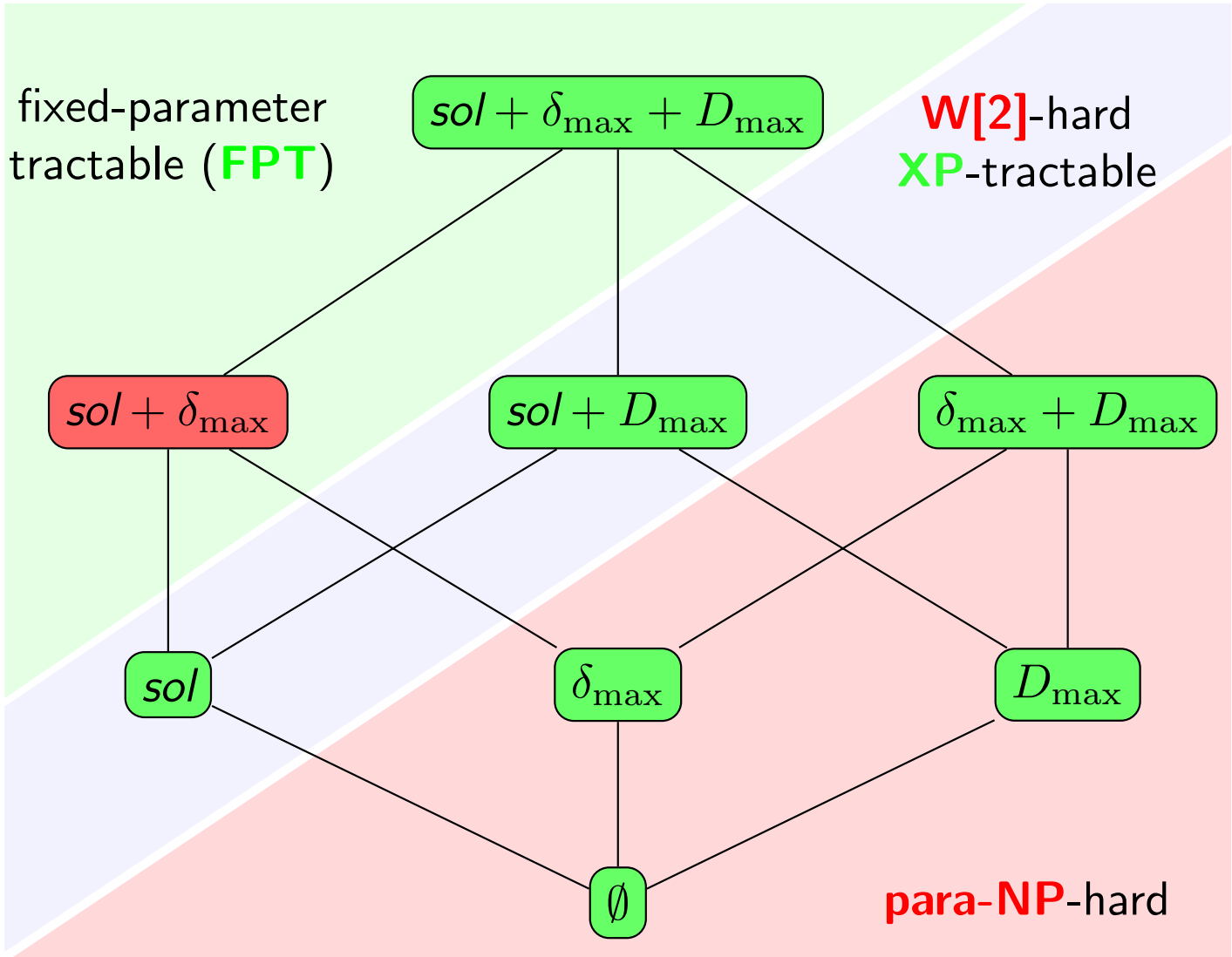


Moreover, $\lambda_{\max}$ can be found using binary search in $\mathcal{O}(\log D_{\max})$ steps!

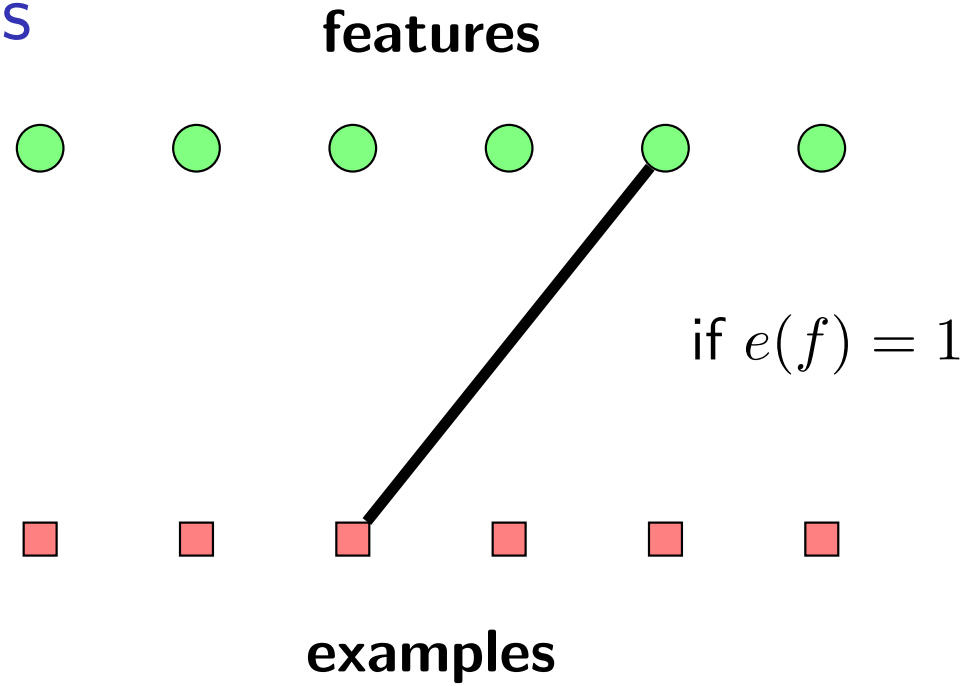
Overview of Results



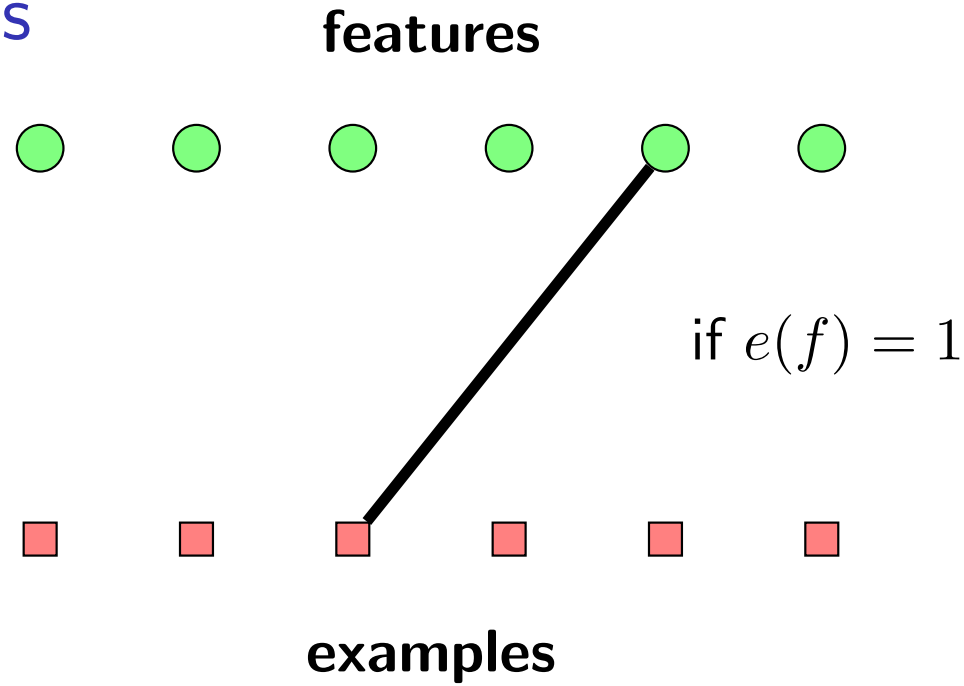
Overview of Results



Further Results

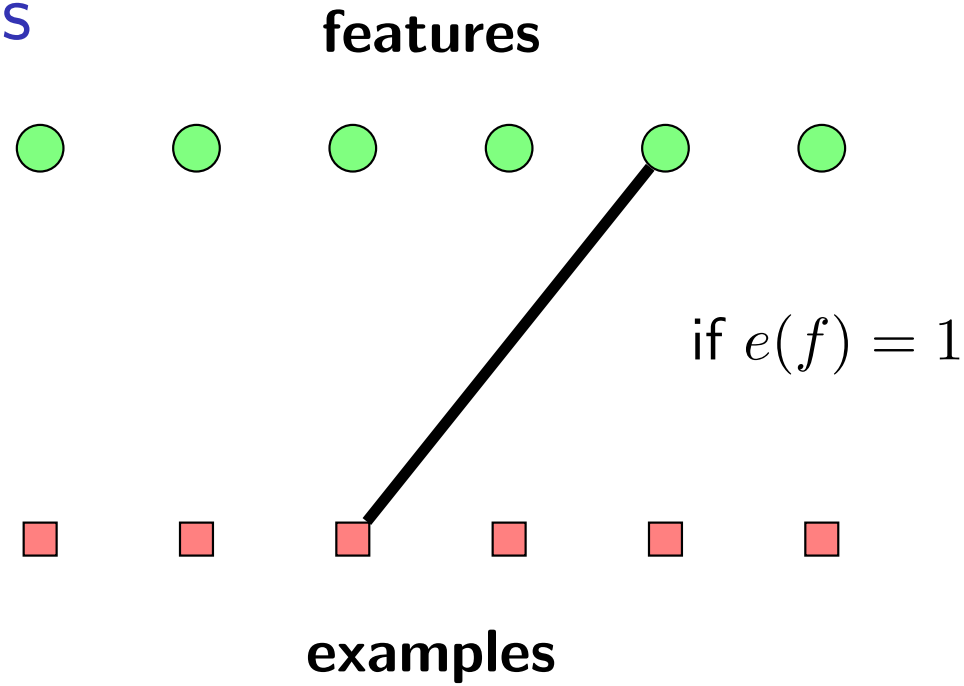


Further Results



renamable rankwidth $\approx$ smallest rankwidth over all possible domain renamings

Further Results

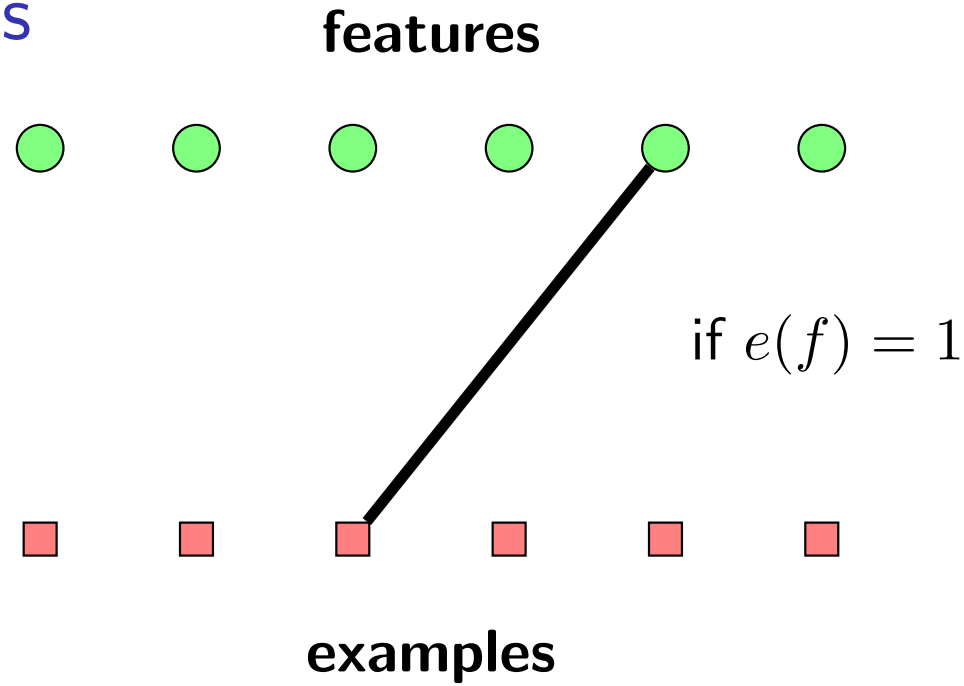


renamable rankwidth $\approx$ smallest rankwidth over all possible domain renamings

Theorem

Minimum Decision Tree Size *is fixed-parameter tractable for* **renamable rankwidth** *(restricted to Boolean CIs).*

Further Results



renamable rankwidth $\approx$ smallest rankwidth over all possible domain renamings

Theorem

Minimum Decision Tree Size *is fixed-parameter tractable for* **renamable rankwidth** *(restricted to Boolean CIs).*

Open

- larger domain?
- minimum depth?

Outlook

- What about Decision Diagrams, DNFs, OBBD's etc.?
- Other structural representations and parameters?
- Applications to Backdoor Trees and Backdoor Cubes for SAT and CSP?

Outlook

- What about Decision Diagrams, DNFs, OBBD's etc.?
- Other structural representations and parameters?
- Applications to Backdoor Trees and Backdoor Cubes for SAT and CSP?

Thank You!