SAT-based Encodings for Optimal Decision Trees with Explicit Paths

Mikoláš Janota[1,2], António Morgado[1]

[1] INESC-ID/IST, Universidade de Lisboa, Portugal
[2] Czech Technical University in Prague, Czech Republic

SAT 2020

Example of Decision Trees

Question: Should I start writing a new article paper ?

Example of Decision Trees

Question: Should I start writing a new article paper ?

Features:

- A - Is the idea for the paper great and innovative?
- B - Are the results bad?
- C - Is there a very close deadline for the paper?

Example of Decision Trees

Question: Should I start writing a new article paper ?

Features:

- A - Is the idea for the paper great and innovative?
- B - Are the results bad?
- C - Is there a very close deadline for the paper?

Samples:

A	B	C	Write?
0	0	0	1
0	0	1	0
0	1	1	0
1	0	1	1
1	0	0	1
1	1	0	0

Example of Decision Trees

Question: Should I start writing a new article paper ?

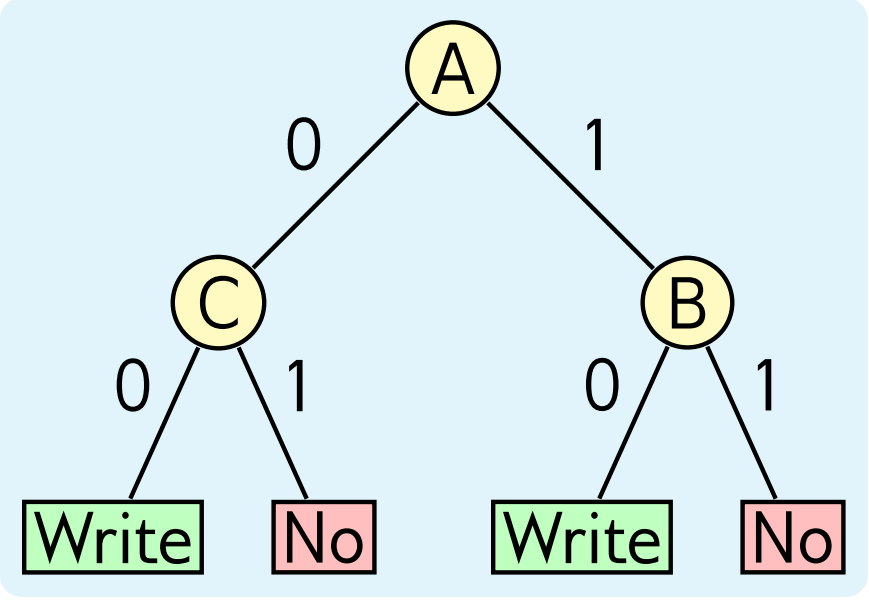
Features:

- A - Is the idea for the paper great and innovative?
- B - Are the results bad?
- C - Is there a very close deadline for the paper?

Samples:

A	B	C	Write?
0	0	0	1
0	0	1	0
0	1	1	0
1	0	1	1
1	0	0	1
1	1	0	0

$\Rightarrow$



Objectives and Motivation

What and Why:

- Given a set of samples, find provably smallest decision tree.

Objectives and Motivation

What and Why:

- Given a set of samples, find provably smallest decision tree.
- Why smallest?
  by Occam's razor, smaller trees generalize better.

Objectives and Motivation

What and Why:

- Given a set of samples, find provably smallest decision tree.
- Why smallest?
  by Occam's razor, smaller trees generalize better.
- Why provably smallest?
  standard algorithms heuristically find small trees

Objectives and Motivation

What and Why:

- Given a set of samples, find provably smallest decision tree.
- Why smallest?
  by Occam's razor, smaller trees generalize better.
- Why provably smallest?
  standard algorithms heuristically find small trees

How:

- Encode into SAT the question is there a tree of size N?

Objectives and Motivation

What and Why:

- Given a set of samples, find provably smallest decision tree.
- Why smallest?
  by Occam's razor, smaller trees generalize better.
- Why provably smallest?
  standard algorithms heuristically find small trees

How:

- Encode into SAT the question is there a tree of size N?
- Look for the smallest tree iteratively.

Objectives and Motivation

What and Why:

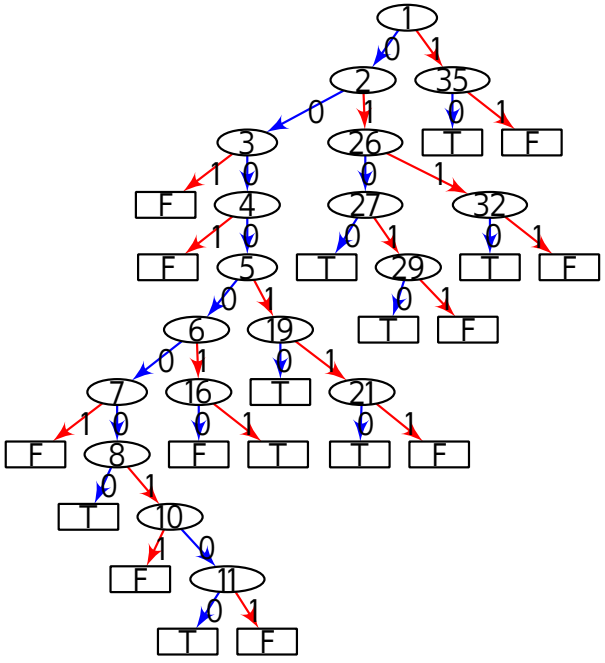
- Given a set of samples, find provably smallest decision tree.
- Why smallest?
  by Occam's razor, smaller trees generalize better.
- Why provably smallest?
  standard algorithms heuristically find small trees

How:

- Encode into SAT the question is there a tree of size N?
- Look for the smallest tree iteratively.
- Two minimization criteria investigated: depth and size.

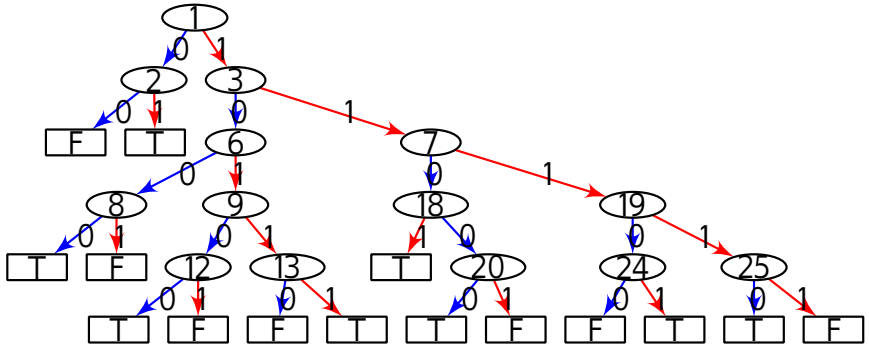
Minimum Depth Optimal Decision Tree

Example benchmark postoperative-patient-data-un 1-un with 50% sampling (approx. 20 features, 40 samples):



`sklearn`

(depth 11, 37 nodes)



`d-dtfinder`

(depth 6, 29 nodes)

Remaining tools timeout (after 1000s)

Encoding Decision Trees into SAT

1. Current SAT approaches:
   - model the tree as DAG
   - impose tree structure

Encoding Decision Trees into SAT

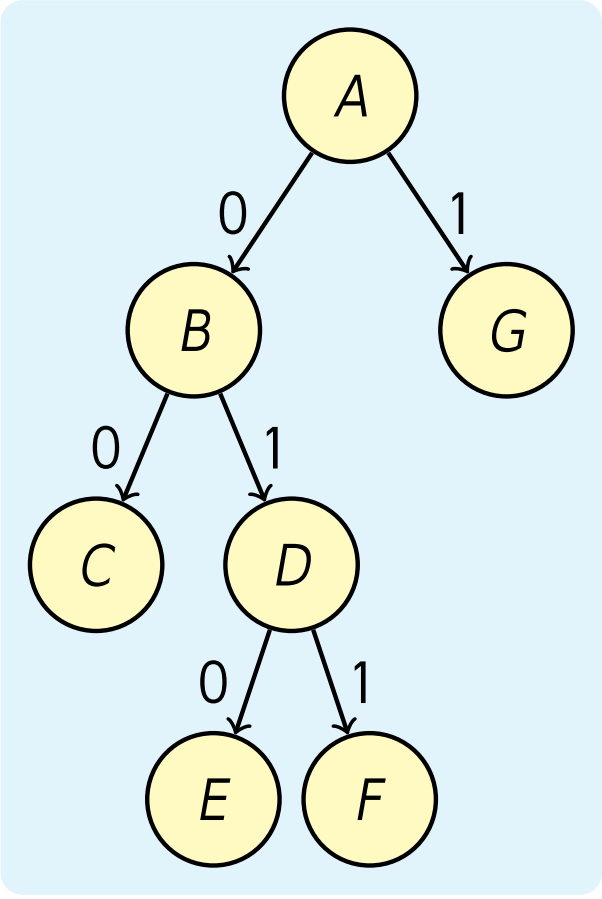
1. Current SAT approaches:
   - model the tree as DAG
   - impose tree structure
2. Our, new, approach:
   - model the tree as a set of paths
   - impose conditions on the paths to make up a (binary) tree

Encoding Decision Trees into SAT

1. Current SAT approaches:
   - model the tree as DAG
   - impose tree structure
2. Our, new, approach:
   - model the tree as a set of paths
   - impose conditions on the paths to make up a (binary) tree
3. Advantages:
   - explicit control over tree's depth
   - no need for cardinality constraints over neighbors (DAG)
   - no need for distinction between internal/leaf nodes

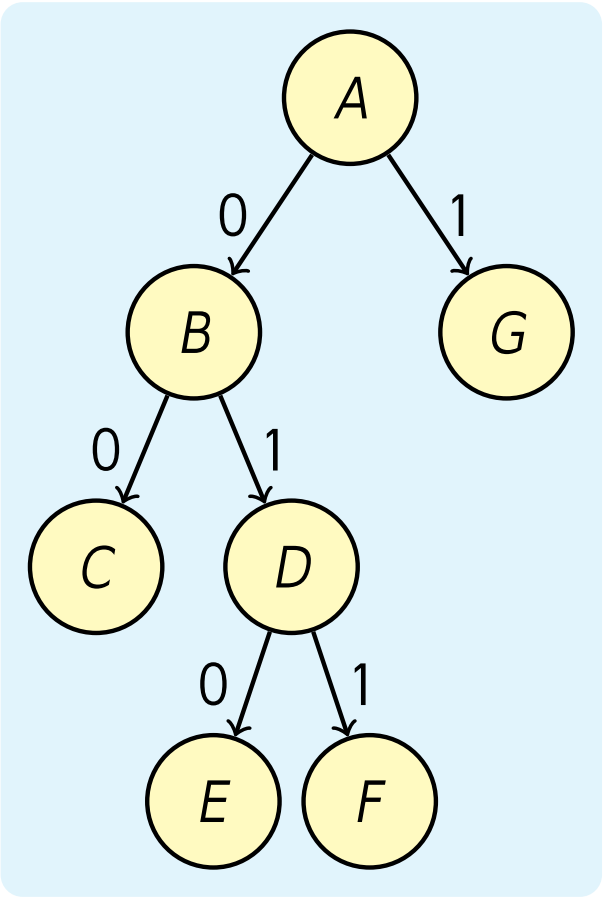
Trees as Paths

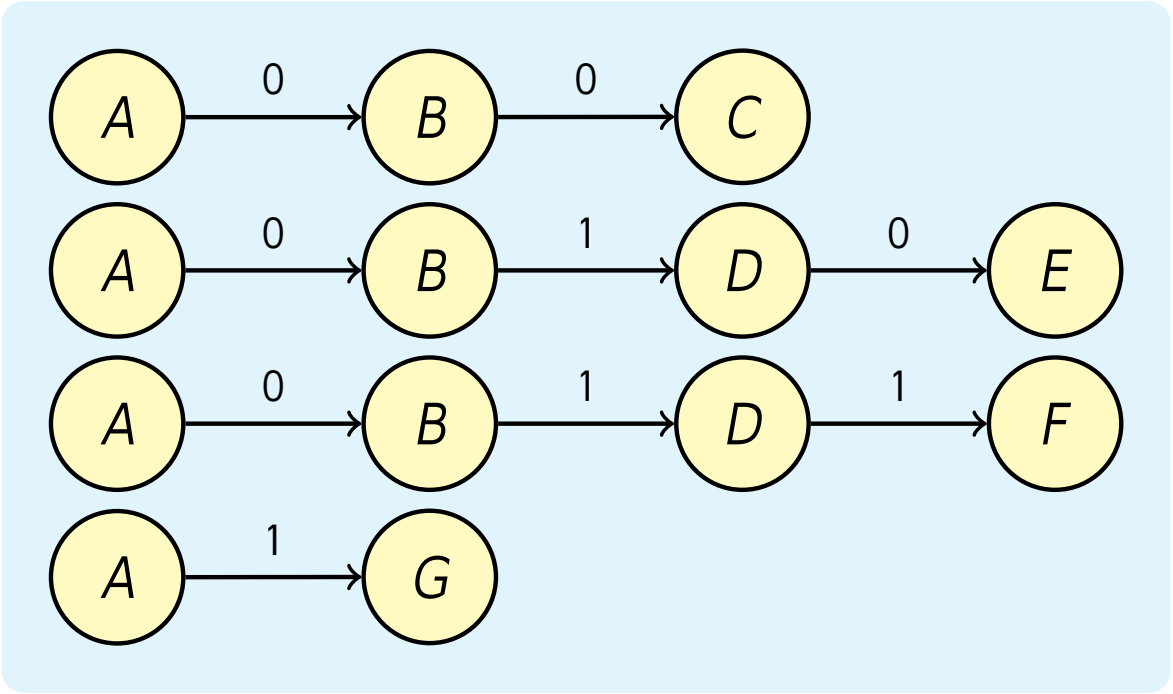
- tree expands to $O(n)$ paths
- consecutive paths overlap until they diverge



Trees as Paths

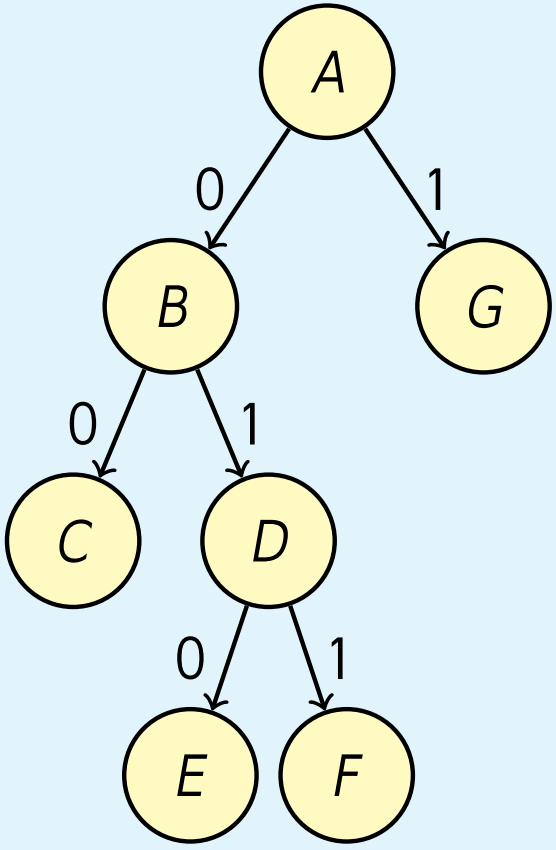
- tree expands to $O(n)$ paths
- consecutive paths overlap until they diverge

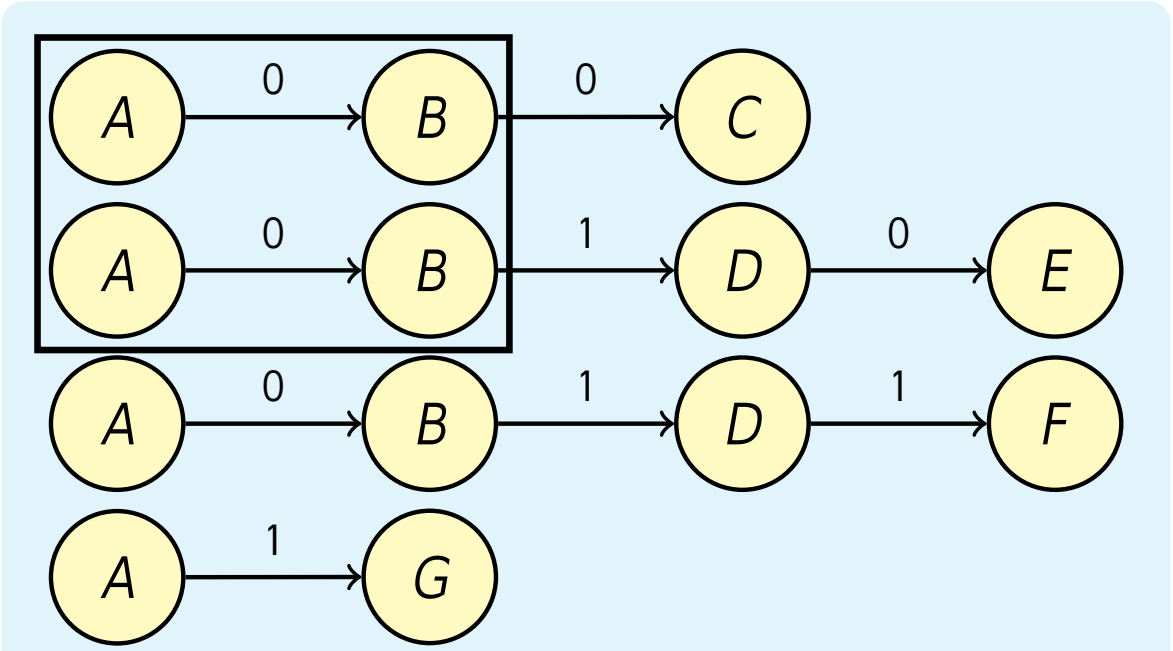




Trees as Paths

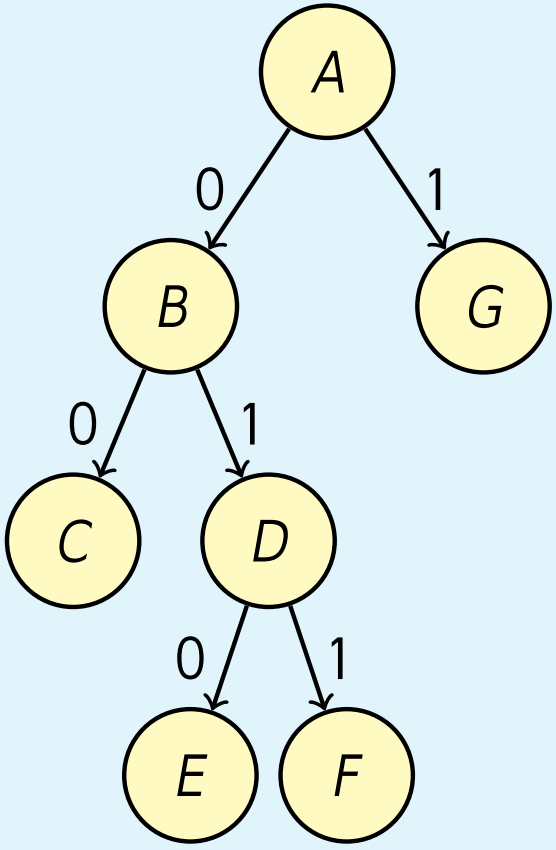
- tree expands to $O(n)$ paths
- consecutive paths overlap until they diverge

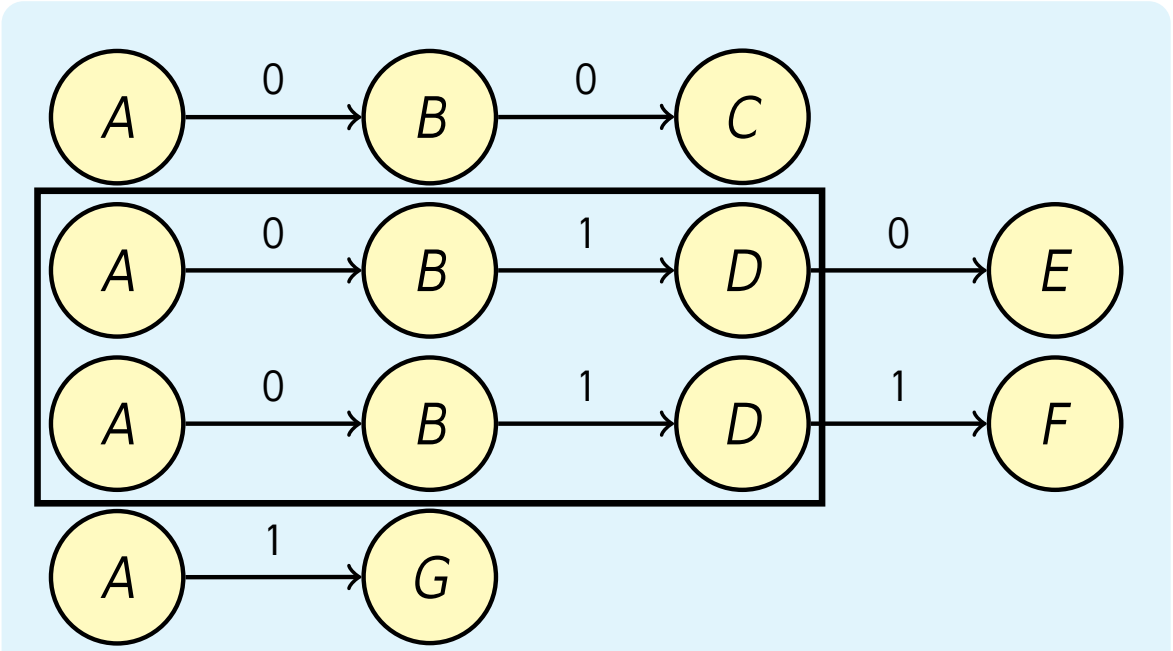




Trees as Paths

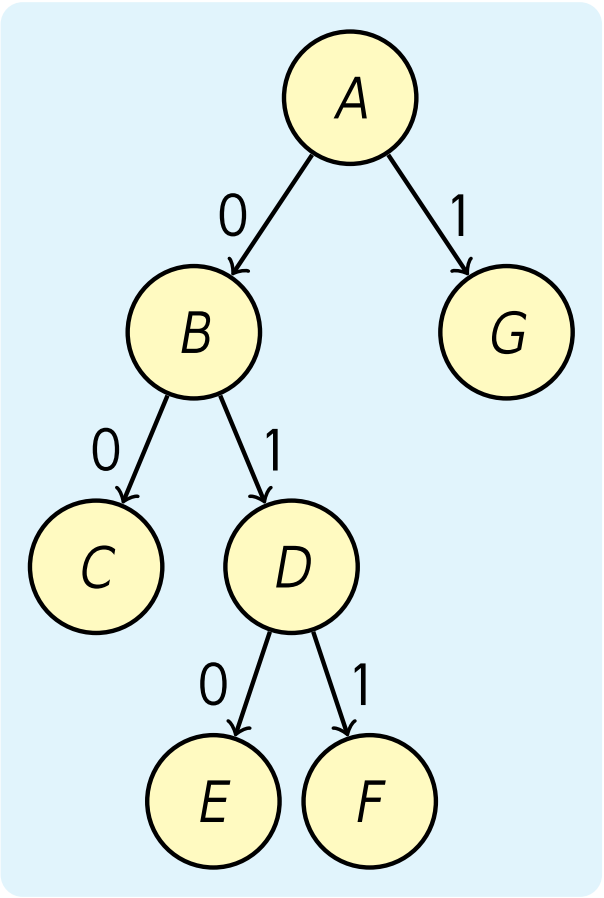
- tree expands to $O(n)$ paths
- consecutive paths overlap until they diverge

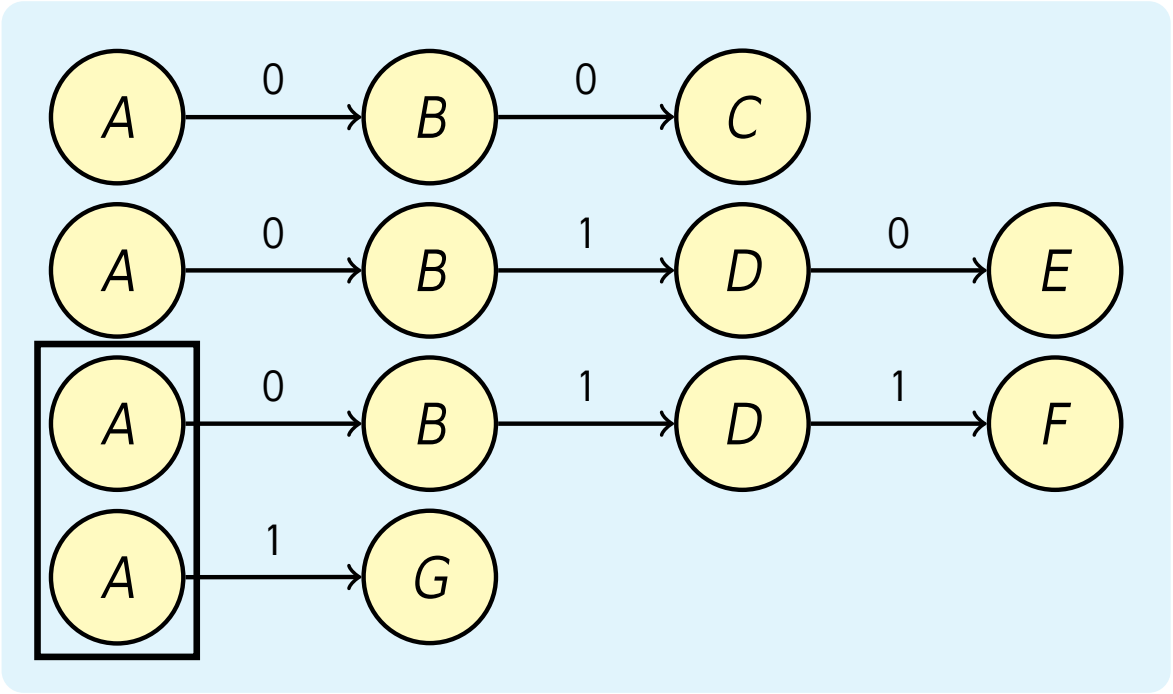




Trees as Paths

- tree expands to $O(n)$ paths
- consecutive paths overlap until they diverge

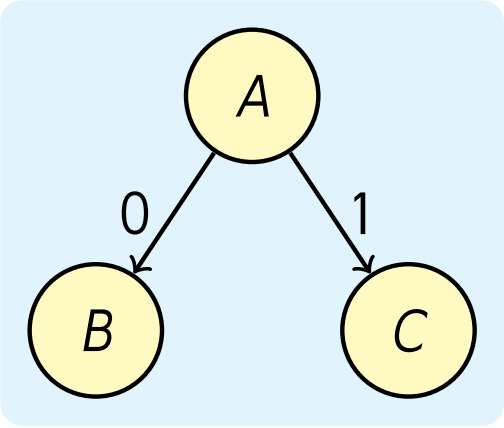




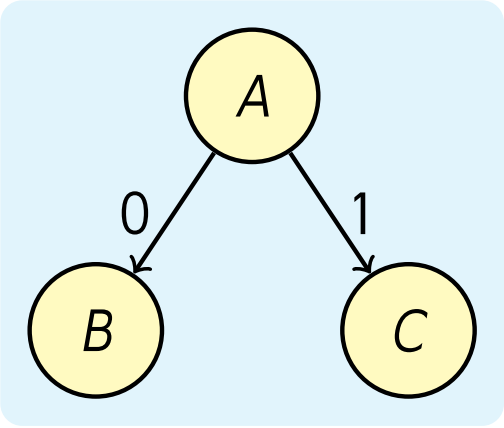
Paths as Booleans

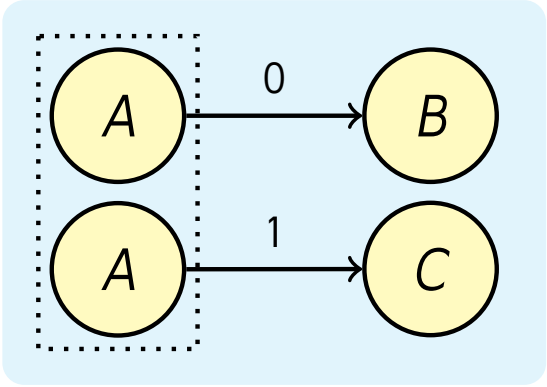
The shape of any path modeled as follows:

Paths as Booleans, example

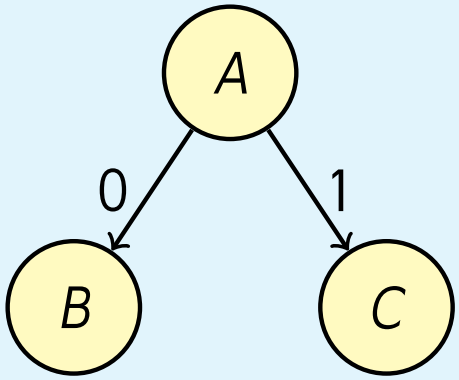


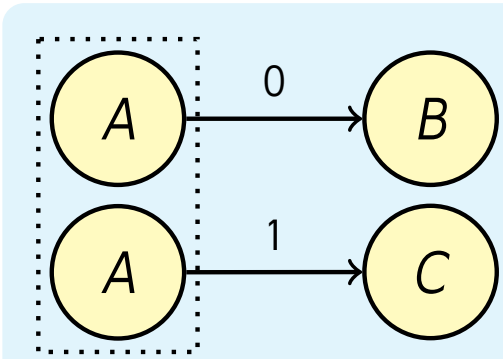
Paths as Booleans, example

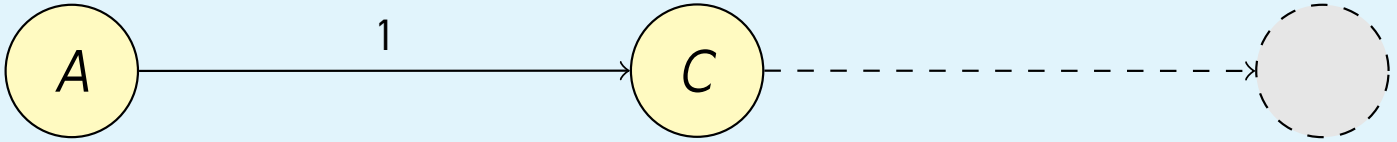




Paths as Booleans, example

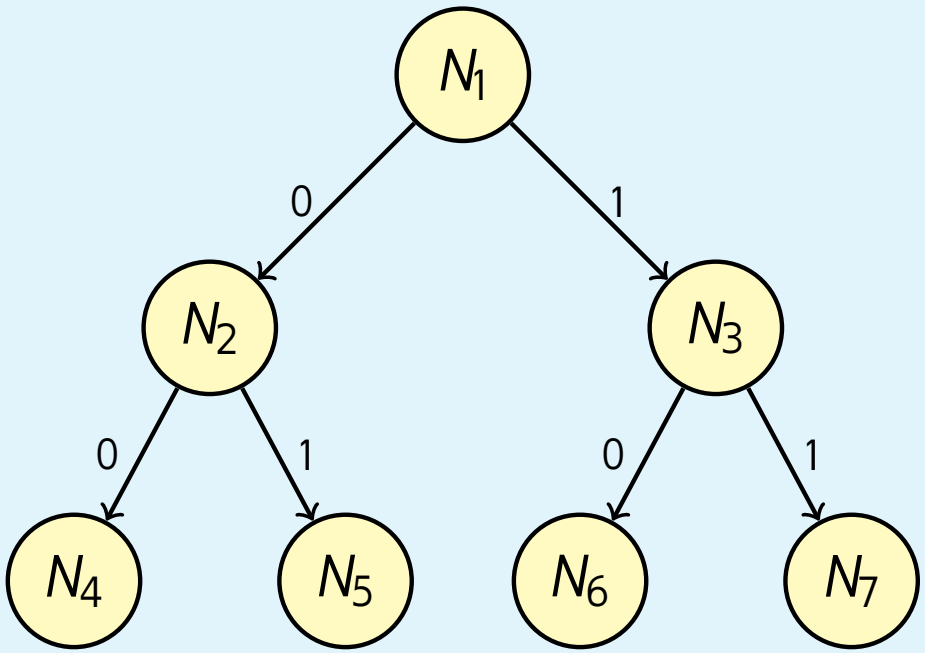






step 1	step 2	step 3
go right=T terminate=F equal to prev=T	go right=* terminate=F equal to prev=F	go right=* terminate=T equal to prev=F

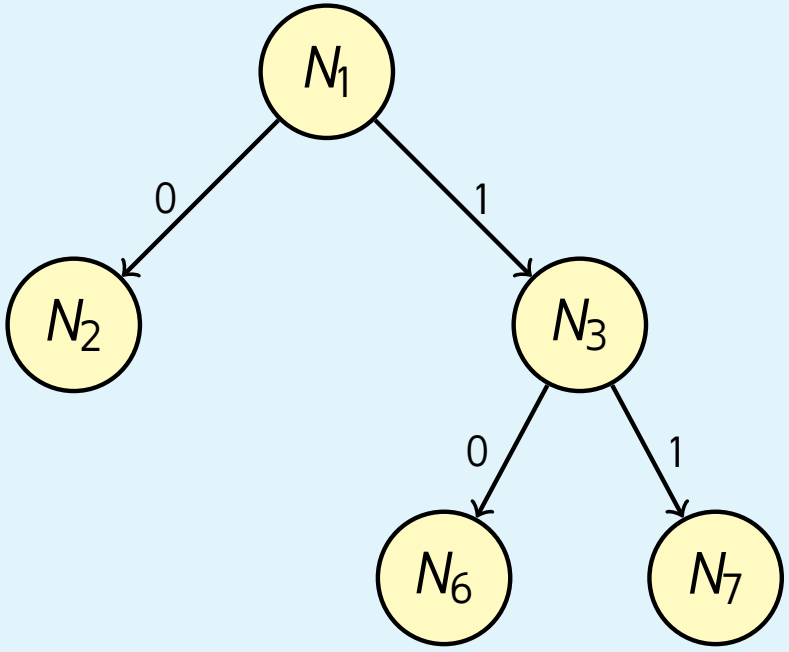
Shaping Paths into Trees



Directions:

0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Shaping Paths into Trees

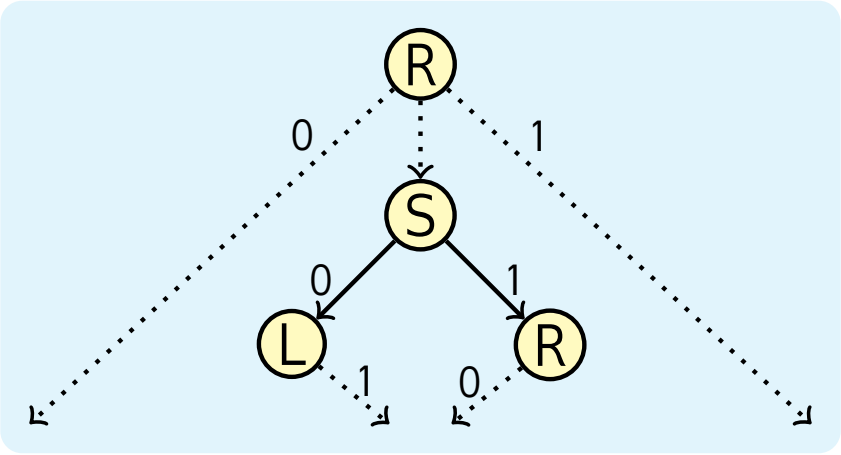


Directions:

0

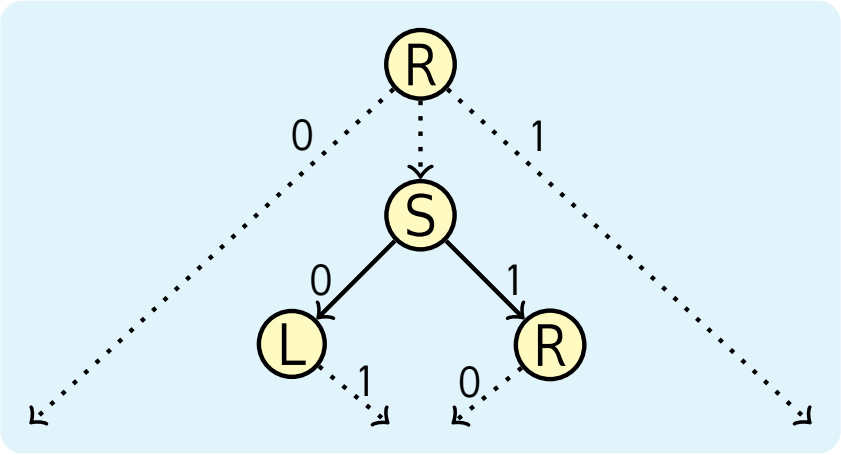
1 0 0
1 0 1
1 1 0
1 1 1

Shaping Paths into Trees, Main Rules



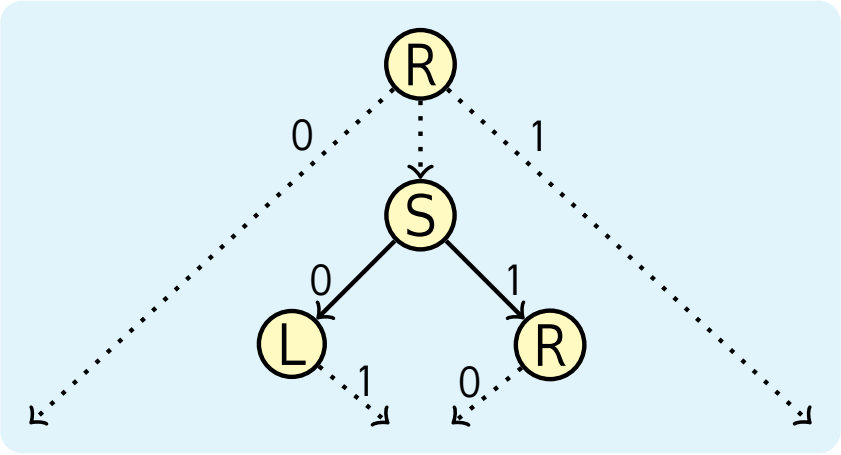
1. First path always goes to the left

Shaping Paths into Trees, Main Rules



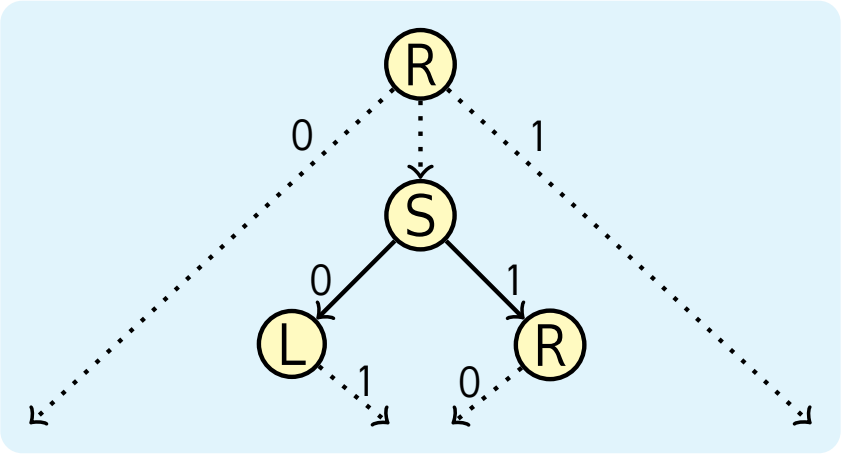
1. First path always goes to the left
2. Last path always goes to the right

Shaping Paths into Trees, Main Rules



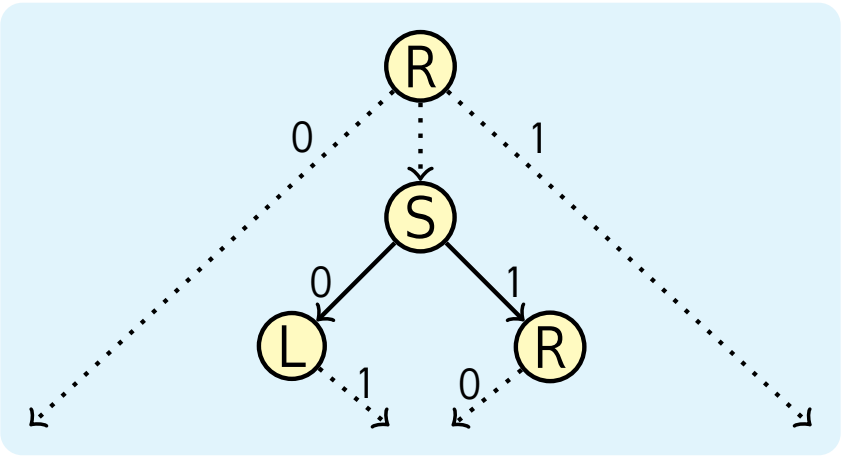
1. First path always goes to the left
2. Last path always goes to the right
3. Paths must not cross.

Shaping Paths into Trees, Main Rules



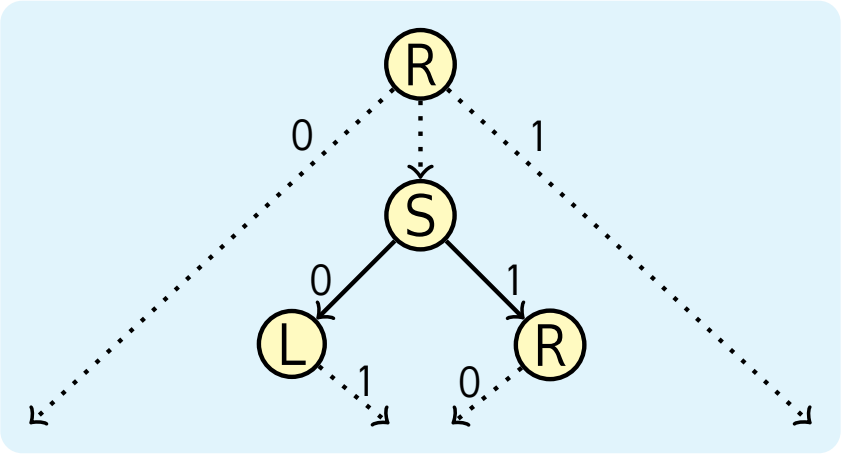
1. First path always goes to the left
2. Last path always goes to the right
3. Paths must not cross.
4. To avoid gaps: once a path diverges . . .

Shaping Paths into Trees, Main Rules



1. First path always goes to the left
2. Last path always goes to the right
3. Paths must not cross.
4. To avoid gaps: once a path diverges . . .
   - it has to keep going left,

Shaping Paths into Trees, Main Rules



1. First path always goes to the left
2. Last path always goes to the right
3. Paths must not cross.
4. To avoid gaps: once a path diverges . . .
   - it has to keep going left,
   - the previous path has to keep going right.

Encoding Semantics

1. Each path has a classification class (single Boolean).

Encoding Semantics

1. Each path has a classification class (single Boolean).
2. Each node is assigned a feature.

Encoding Semantics

1. Each path has a classification class (single Boolean).
2. Each node is assigned a feature.
3. For each path determine which samples reach its end.

Encoding Semantics

1. Each path has a classification class (single Boolean).
2. Each node is assigned a feature.
3. For each path determine which samples reach its end.
4. If a positive sample reaches the end of a path, the path must be positive.

Encoding Semantics

1. Each path has a classification class (single Boolean).
2. Each node is assigned a feature.
3. For each path determine which samples reach its end.
4. If a positive sample reaches the end of a path, the path must be positive.
5. If a negative sample reaches the end of a path, the path must be negative.

Optimizations

- Enforcing example matching

Optimizations

- Enforcing example matching
- Pure features

Optimizations

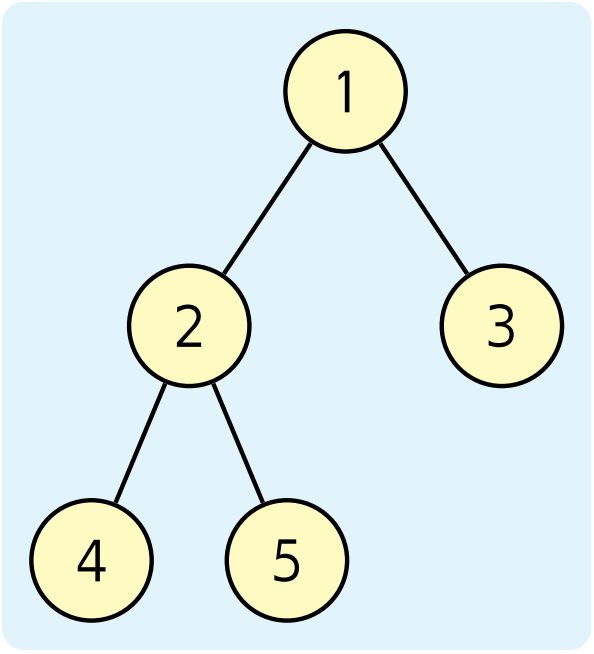
- Enforcing example matching
- Pure features
- Quasi-pure features

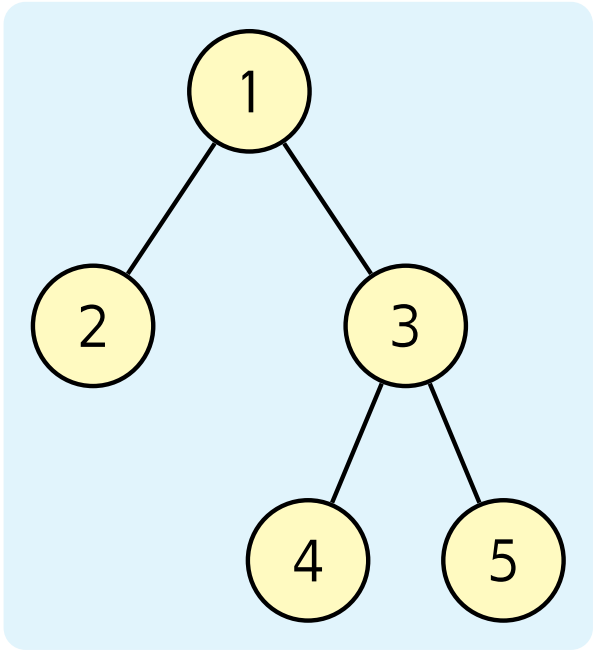
Optimizations

- Enforcing example matching
- Pure features
- Quasi-pure features
- Path lower/upper bounds

Topologies

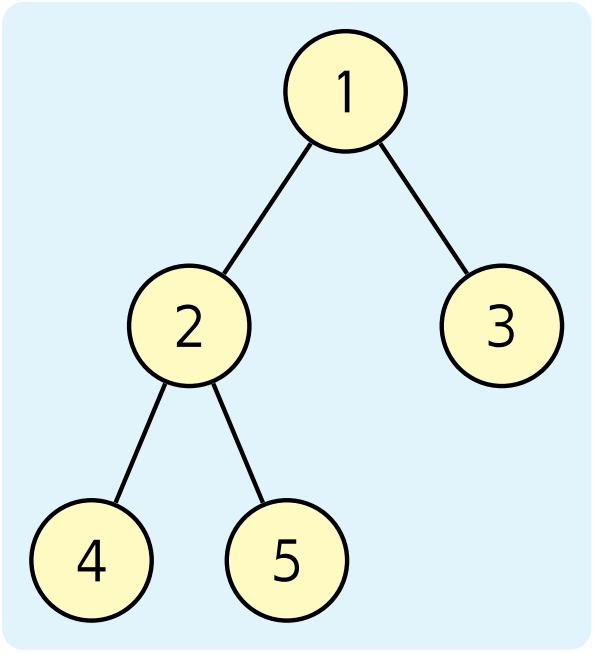
- Observation: number of topologies for small trees is low

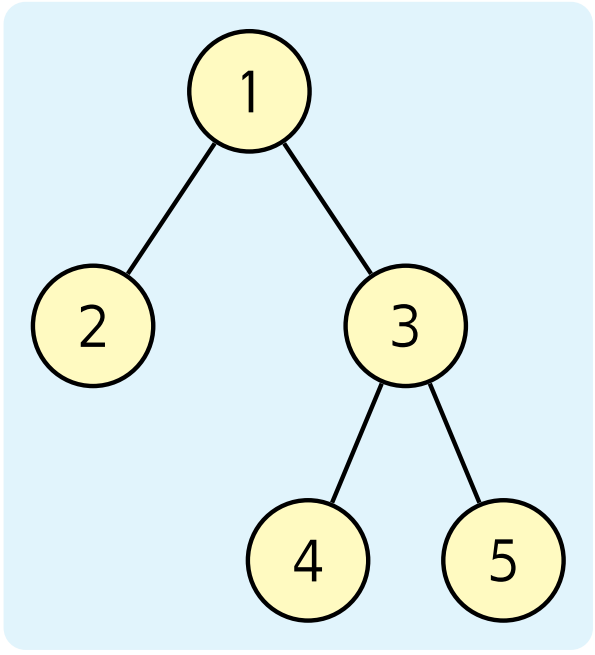




Topologies

- Observation: number of topologies for small trees is low





#**n**	5	7	9	11	13	15	17	19	21	...
#**t**	2	5	14	42	132	429	1,430	4,862	16,796	...

Enumerating Topologies

- Split the search space for each topology
- . . . the solver only fills in features and categories.
- Essentially *cube and conquer*,
  but informed by the problem.

Iterative SAT Calls

- going from smaller to bigger, i.e. UNSAT to SAT

Iterative SAT Calls

- going from smaller to bigger, i.e. UNSAT to SAT
- either iterate over depth and size or over size

Iterative SAT Calls

- going from smaller to bigger, i.e. UNSAT to SAT
- either iterate over depth and size or over size
- topology enumeration combines iterative and non-iterative SAT calls

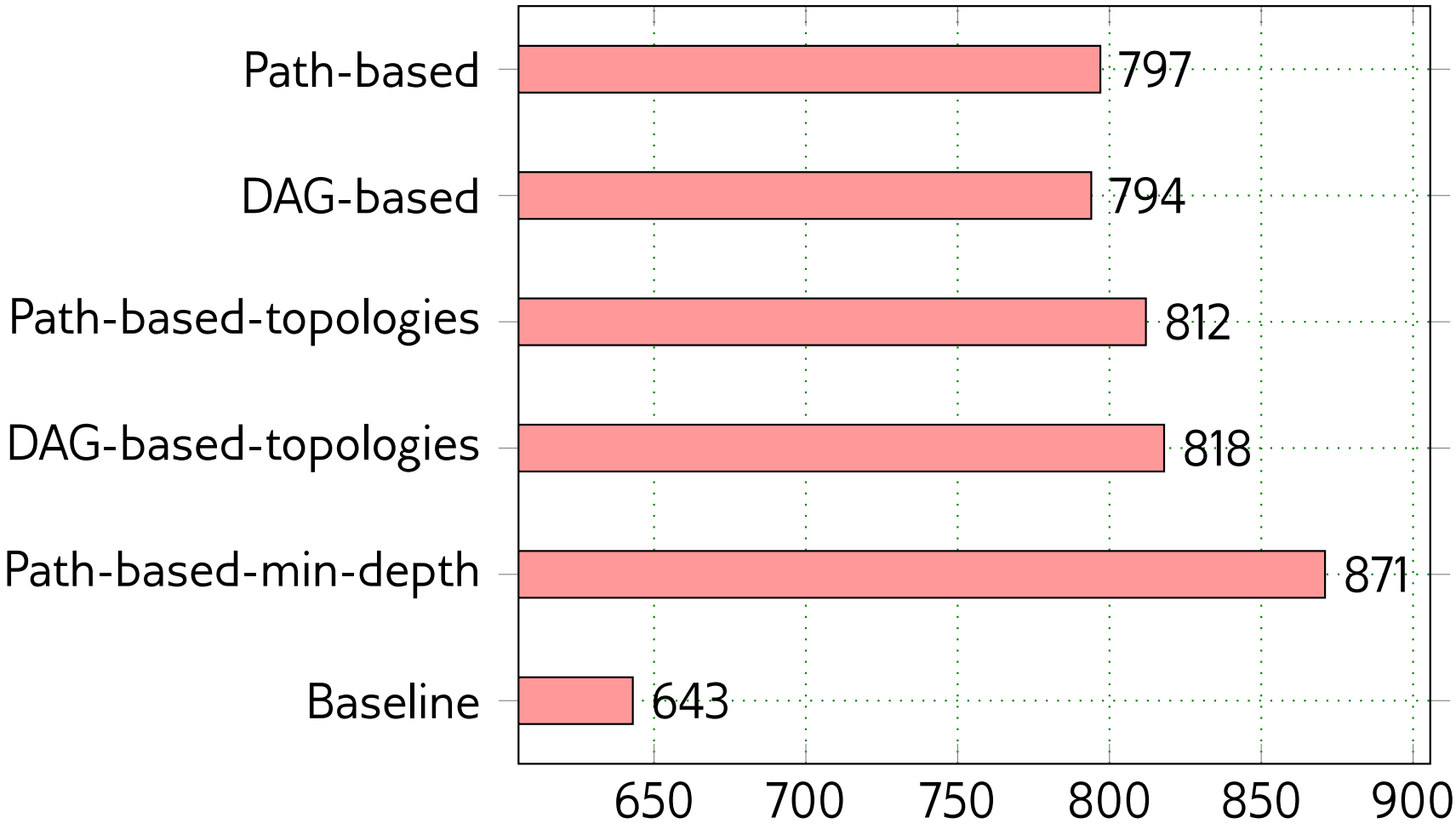
Iterative SAT Calls

- going from smaller to bigger, i.e. UNSAT to SAT
- either iterate over depth and size or over size
- topology enumeration combines iterative and non-iterative SAT calls
- partial topologies supported for larger numbers

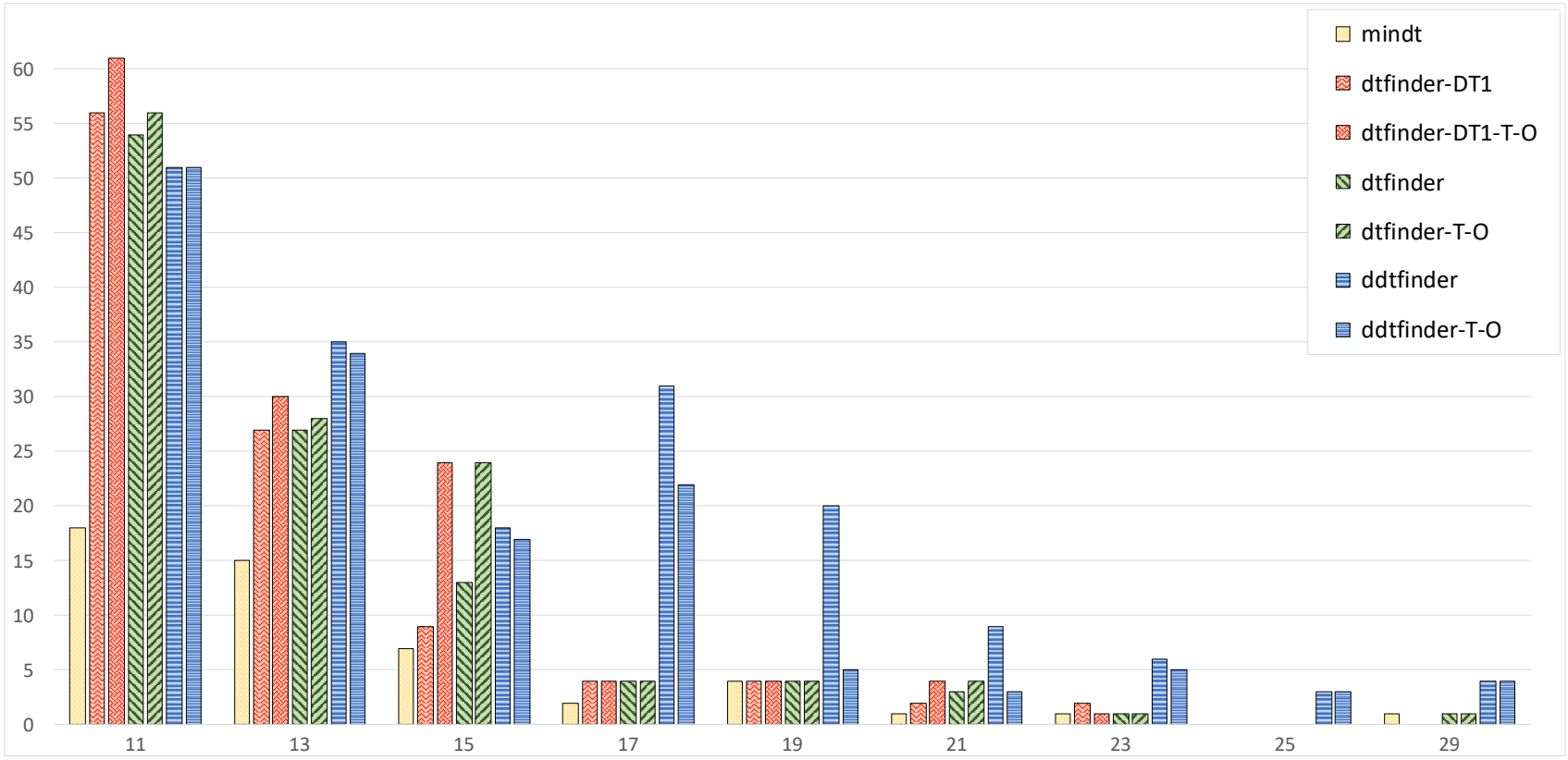
Iterative SAT Calls

- going from smaller to bigger, i.e. UNSAT to SAT
- either iterate over depth and size or over size
- topology enumeration combines iterative and non-iterative SAT calls
- partial topologies supported for larger numbers
- heuristics for topology order also explored

Summary of Results



Distribution of tree sizes



Conclusions and Future Work

- Novel SAT-based encoding for decision trees,
  enables natively controlling both the tree's size and depth.

Conclusions and Future Work

- Novel SAT-based encoding for decision trees,
  enables natively controlling both the tree's size and depth.
- Search-space splitting by topology enumeration.

Conclusions and Future Work

- Novel SAT-based encoding for decision trees,
  enables natively controlling both the tree's size and depth.
- Search-space splitting by topology enumeration.
- Our implementation outperforms existing work.

Conclusions and Future Work

- Novel SAT-based encoding for decision trees,
  enables natively controlling both the tree's size and depth.
- Search-space splitting by topology enumeration.
- Our implementation outperforms existing work.
- Depth-minimization allows optimizing larger instances.

Conclusions and Future Work

- Novel SAT-based encoding for decision trees,
  enables natively controlling both the tree's size and depth.
- Search-space splitting by topology enumeration.
- Our implementation outperforms existing work.
- Depth-minimization allows optimizing larger instances.
- Integrate the proposed techniques into more expressive approaches (e.g. SMT-based synthesis)

Conclusions and Future Work

- Novel SAT-based encoding for decision trees,
  enables natively controlling both the tree's size and depth.
- Search-space splitting by topology enumeration.
- Our implementation outperforms existing work.
- Depth-minimization allows optimizing larger instances.
- Integrate the proposed techniques into more expressive approaches (e.g. SMT-based synthesis)
- Integrate our tool with greedy approaches, for example:

Conclusions and Future Work

- Novel SAT-based encoding for decision trees,
  enables natively controlling both the tree's size and depth.
- Search-space splitting by topology enumeration.
- Our implementation outperforms existing work.
- Depth-minimization allows optimizing larger instances.
- Integrate the proposed techniques into more expressive approaches (e.g. SMT-based synthesis)
- Integrate our tool with greedy approaches, for example:
  - hybrid between a greedy approach and an exact.

Conclusions and Future Work

- Novel SAT-based encoding for decision trees,
  enables natively controlling both the tree's size and depth.
- Search-space splitting by topology enumeration.
- Our implementation outperforms existing work.
- Depth-minimization allows optimizing larger instances.
- Integrate the proposed techniques into more expressive approaches (e.g. SMT-based synthesis)
- Integrate our tool with greedy approaches, for example:
  - hybrid between a greedy approach and an exact.
  - ensembles, where only limited depth is considered

Thank you for your attention!

Questions?