Width-Parameterized SAT

Time-Space Tradeoffs

Eric Allender[1] Shiteng Chen[2] Tiancheng Lou[2]
Periklis Papakonstantinou[2] Bangsheng Tang[2]

[1]Rutgers University

[2]IIIS, Tsinghua University

October 10, 2011

Satisfiability Problem

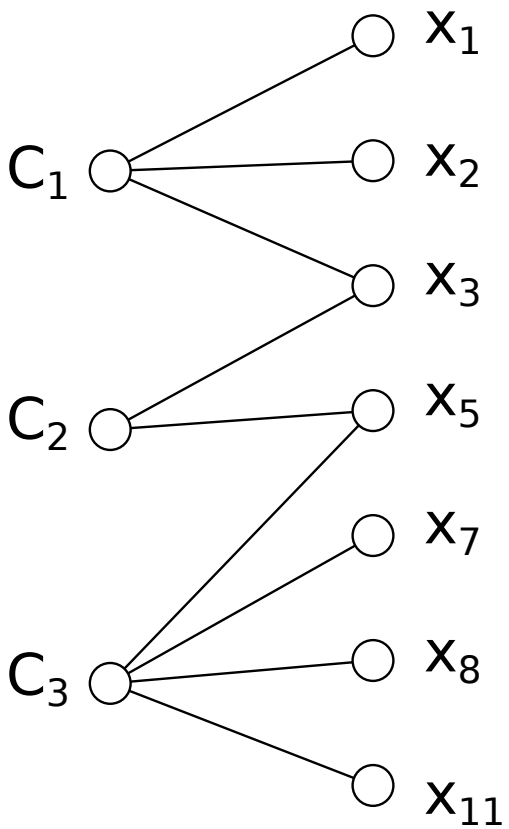
SAT: satisfiability of a **CNF** ϕ

- $(x_1 \vee x_2 \vee \overline{x_3}) \wedge (x_3 \vee \overline{x_5}) \wedge (\overline{x_1} \vee x_7 \vee x_8 \vee \overline{x_{11}})$
- k-**SAT**: each clause has $\leq k$ literals
- Prototypical **NP**-complete problem
- Extensively studied in theoretical and empirical works

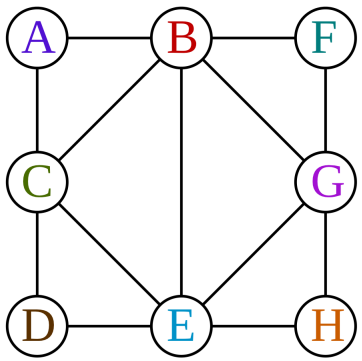
This work, **SAT** with small width-parameters

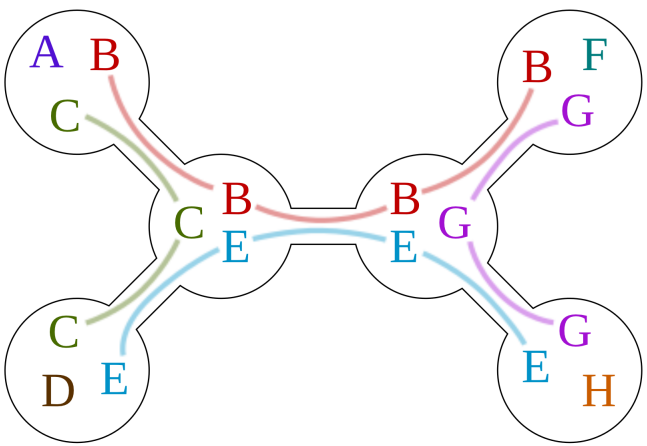
Incidence Graph G_ϕ

$$\phi = \underbrace{(x_1 \vee x_2 \vee \overline{x_3})}_{C_1} \wedge \underbrace{(x_3 \vee \overline{x_5})}_{C_2} \wedge \underbrace{(\overline{x_1} \vee x_7 \vee x_8 \vee \overline{x_{11}})}_{C_3}$$



Tree Width





- Tree Decomposition
  - Nodes $\rightarrow$ Nodes of a *tree*
- $\mathcal{TW}(G)$
  - minimum size of biggest node minus 1
  - smaller $\rightarrow$ the more like a tree
  - $\mathcal{PW}(G)$: when the decomposition is a *path*
- $\mathcal{TW}(G) \leq \mathcal{PW}(G) \leq \log|V| \cdot \mathcal{TW}(G)$.
- $\mathcal{TW}(\phi) = \mathcal{TW}(G_\phi)$

Bounded Tree-width **SAT**

SAT$_{\text{tw}}(w(n))$

- Input: ϕ, TD($\#vars + \#clauses \leq n, \mathcal{TW}(\phi) \leq w(n)$)
- e.g. $w(n) = \log^2 n$
- [AR02]
  - in time $O^*(2^{O(w(n))})$ and space $O^*(2^{O(w(n))})$
  - *in time $O^*(2^{O(w(n))})$ and space $O^*(1)$?*
- [GP08]
  - in time $O^*(2^{O(w(n)\log n)})$ and space $O^*(1)$
  - non-explicit, only works for k-**SAT** version

Bounded Tree-width **SAT**

SAT$_{\text{tw}}(w(n))$

- Input: ϕ, TD($\#vars + \#clauses \leq n, \mathcal{TW}(\phi) \leq w(n)$)
- e.g. $w(n) = \log^2 n$
- [AR02]
  - in time $O^*(2^{O(w(n))})$ and space $O^*(2^{O(w(n))})$
  - *in time* $O^*(2^{O(w(n))})$ *and space* $O^*(1)$*?*
- [GP08]
  - in time $O^*(2^{O(w(n)\log n)})$ and space $O^*(1)$
  - non-explicit, only works for k-**SAT** version

Bounded Tree-width **SAT**

SAT$_{\text{tw}}(w(n))$

- Input: ϕ, TD($\#vars + \#clauses \leq n, \mathcal{TW}(\phi) \leq w(n)$)
- e.g. $w(n) = \log^2 n$
- [AR02]
  - in time $O^*(2^{O(w(n))})$ and space $O^*(2^{O(w(n))})$
  - *in time* $O^*(2^{O(w(n))})$ *and space* $O^*(1)$*?*
- [GP08]
  - in time $O^*(2^{O(w(n)\log n)})$ and space $O^*(1)$
  - non-explicit, only works for k-**SAT** version

Bounded Path-width **SAT**

$\mathbf{SAT}_{\mathrm{pw}}(w(n))$

- special case of $\mathbf{SAT}_{\mathrm{tw}}(w(n))$
- [Pap09]
  - $\mathbf{SAT}_{\mathrm{pw}}(\log^k n)$ is complete for $\mathbf{NL}[\log^{k-1} n]$.
  - no similar characterization for $\mathbf{SAT}_{\mathrm{tw}}(\log^k n)$

Our Contributions

Algorithmic Results

- Two explicit algorithms for $\mathbf{SAT}_{\mathrm{tw}}$:
  - time-efficient
  - space-efficient
- A trade-off algorithm
- A Conjecture

Complexity Theoretic Results

- New Characterizations of $\mathbf{SAT}_{\mathrm{tw}}$, $\mathbf{SAT}_{\mathrm{pw}}$
- $\mathbf{SAT}_{\mathrm{pw}}(w(n))$ easier than $\mathbf{SAT}_{\mathrm{tw}}(w(n))$
- Our conjecture $\Rightarrow$ something we believe

Our Contributions

Algorithmic Results

- Two explicit algorithms for $\mathbf{SAT}_{\mathrm{tw}}$:
  - time-efficient
  - space-efficient
- A trade-off algorithm
- A Conjecture

Complexity Theoretic Results

- New Characterizations of $\mathbf{SAT}_{\mathrm{tw}}$, $\mathbf{SAT}_{\mathrm{pw}}$
- $\mathbf{SAT}_{\mathrm{pw}}(w(n))$ easier than $\mathbf{SAT}_{\mathrm{tw}}(w(n))$
- Our conjecture $\Rightarrow$ something we believe

Our Contributions

Algorithmic Results

- Two explicit algorithms for $\mathbf{SAT}_{\mathrm{tw}}$:
  - time-efficient
  - space-efficient
- A trade-off algorithm
- A Conjecture

Complexity Theoretic Results

- New Characterizations of $\mathbf{SAT}_{\mathrm{tw}}$, $\mathbf{SAT}_{\mathrm{pw}}$
- $\mathbf{SAT}_{\mathrm{pw}}(w(n))$ easier than $\mathbf{SAT}_{\mathrm{tw}}(w(n))$
- Our conjecture $\Rightarrow$ something we believe

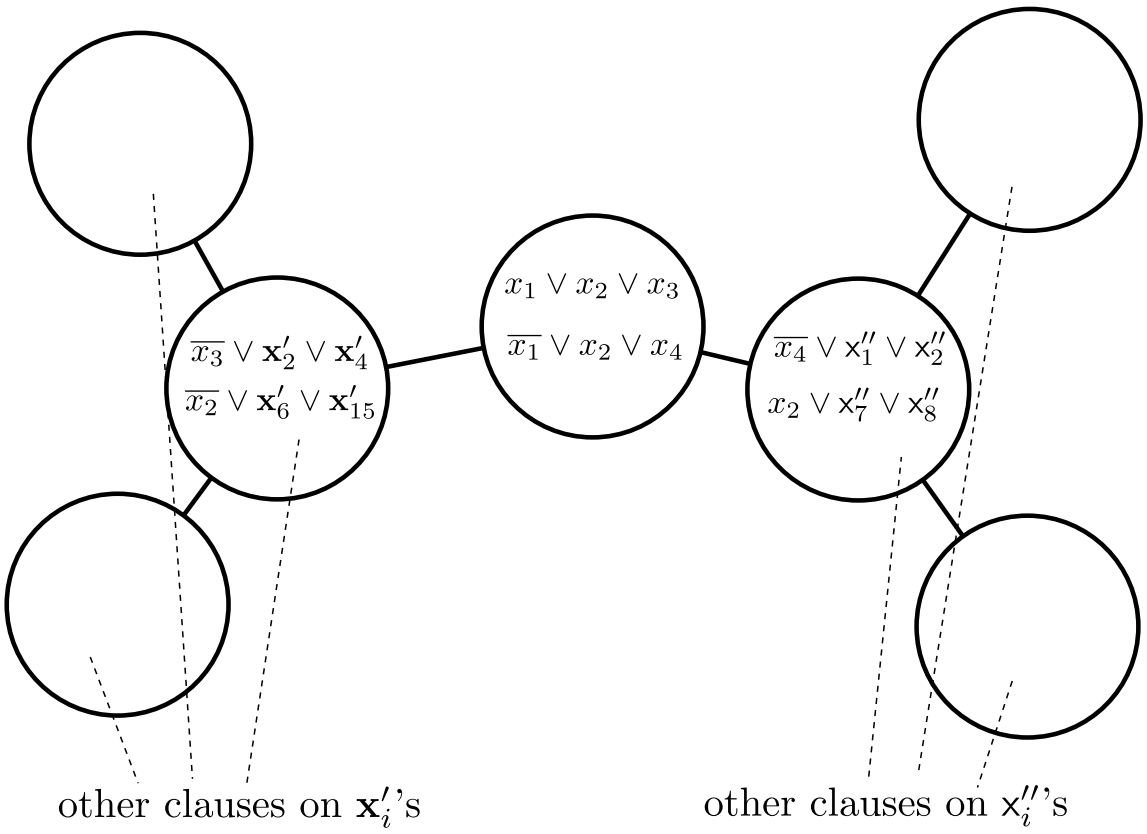
Assumptions on the Decomposition

1. # of nodes is $O(w(n)n)$
2. The tree is of bounded degree
3. If a clause appears in one node, then all its variables also appear in the same node.

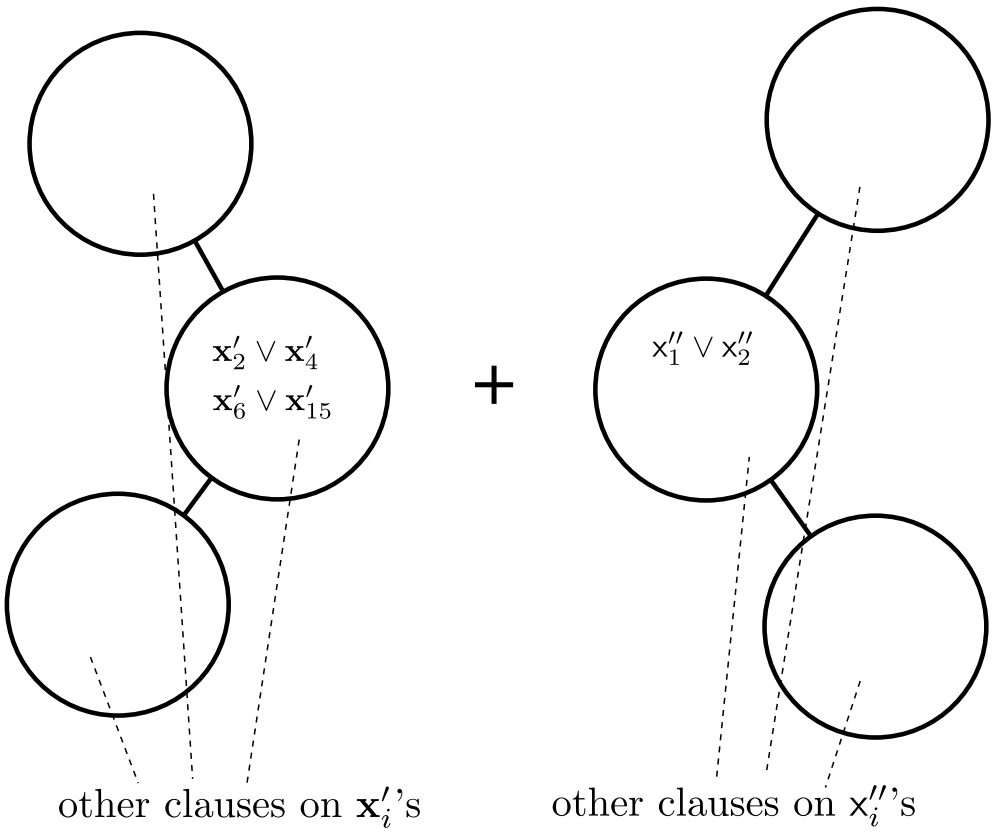
Assumptions on the Decomposition

1. # of nodes is $O(w(n)n)$
2. The tree is of bounded degree
3. If a clause appears in one node, then all its variables also appear in the same node.

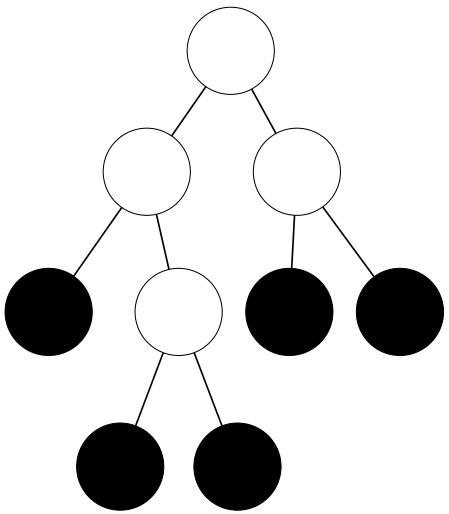
Basic Property



Basic Property



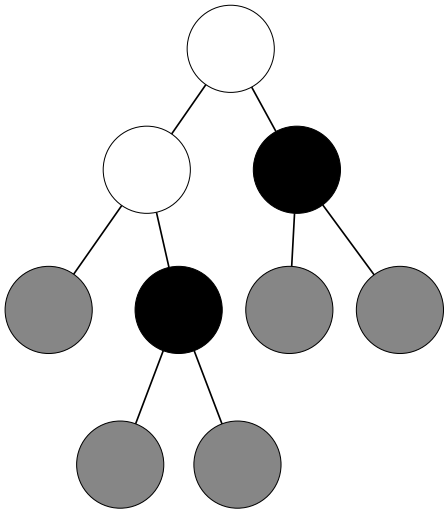
Time-efficient



- At each node
  - $2^{w(n)}$-size boolean array
  - record those that can be extended to satisfying assignments for the whole subtree rooted at this it.
- Bottom-Up
- Consistency test
  - Adjacent nodes

$\mathbf{SAT}_{\mathrm{tw}}(w(n))$ can be done in time $O^*(2^{2w(n)})$, space $O^*(2^{w(n)})$

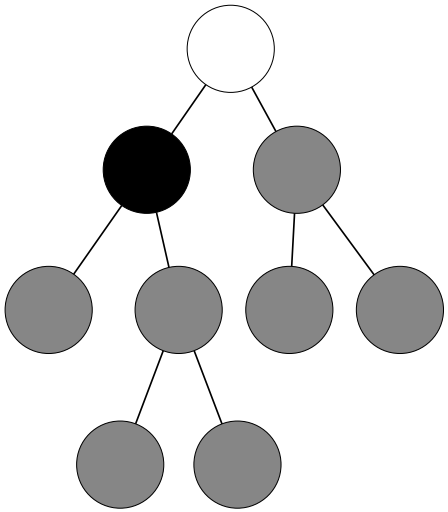
Time-efficient



- At each node
  - $2^{w(n)}$-size boolean array
  - record those that can be extended to satisfying assignments for the whole subtree rooted at this it.
- Bottom-Up
- Consistency test
  - Adjacent nodes

$\mathbf{SAT}_{\mathrm{tw}}(w(n))$ can be done in time $O^*(2^{2w(n)})$, space $O^*(2^{w(n)})$

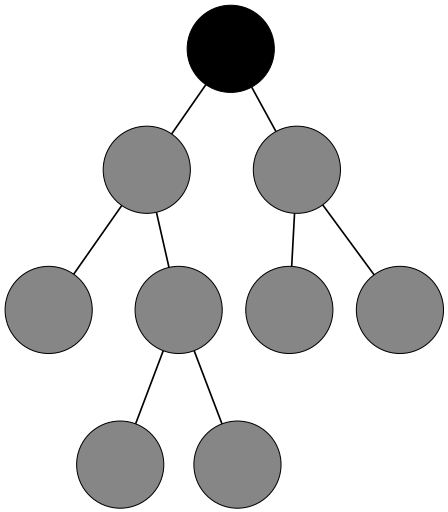
Time-efficient



- At each node
  - $2^{w(n)}$-size boolean array
  - record those that can be extended to satisfying assignments for the whole subtree rooted at this it.
- Bottom-Up
- Consistency test
  - Adjacent nodes

$\mathbf{SAT}_{\mathrm{tw}}(w(n))$ can be done in time $O^*(2^{2w(n)})$, space $O^*(2^{w(n)})$

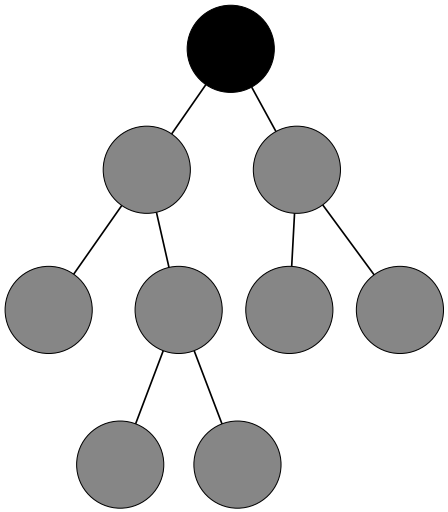
Time-efficient



- At each node
  - $2^{w(n)}$-size boolean array
  - record those that can be extended to satisfying assignments for the whole subtree rooted at this it.
- Bottom-Up
- Consistency test
  - Adjacent nodes

$\mathbf{SAT}_{\mathrm{tw}}(w(n))$ can be done in time $O^*(2^{2w(n)})$, space $O^*(2^{w(n)})$

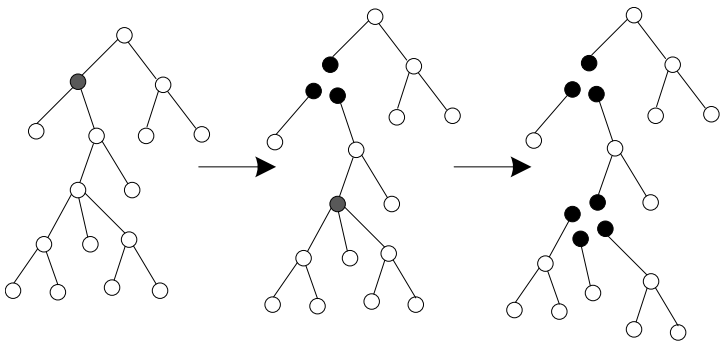
Time-efficient



- At each node
  - $2^{w(n)}$-size boolean array
  - record those that can be extended to satisfying assignments for the whole subtree rooted at this it.
- Bottom-Up
- Consistency test
  - Adjacent nodes

$\mathbf{SAT}_{\mathrm{tw}}(w(n))$ can be done in time $O^*(2^{2w(n)})$, space $O^*(2^{w(n)})$

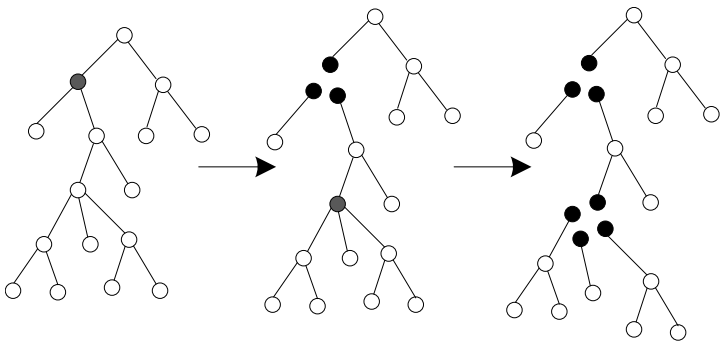
Space-efficient



1. Find a *balanced* splitting node v
2. Fix an assignment to v, consistent with all the existing splitting nodes
3. Split and recurse into subtrees

$\mathbf{SAT}_{\mathrm{tw}}(w(n))$ can be done in time $O^*(2^{w(n)\log n})$, space $O^*(1)$

Space-efficient



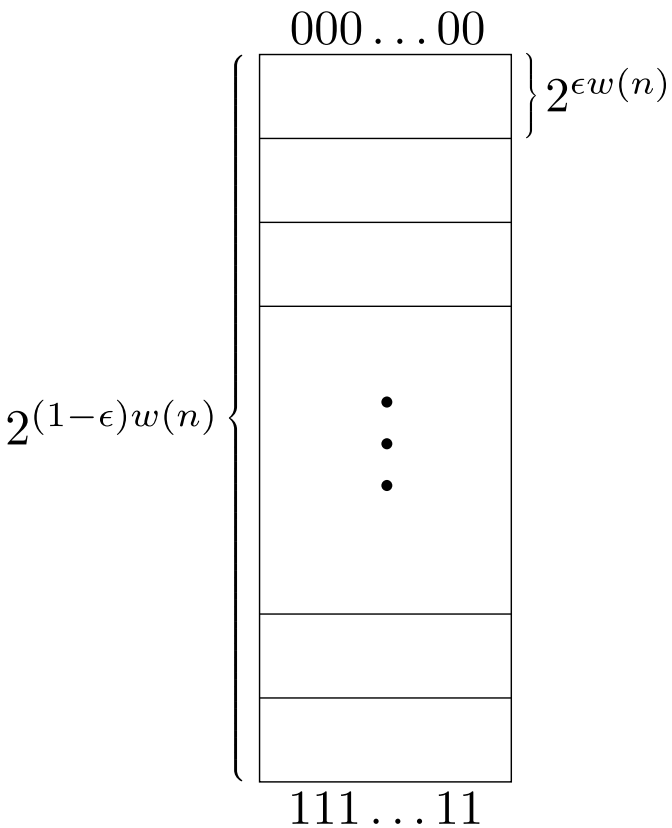
1. Find a *balanced* splitting node v
2. Fix an assignment to v, consistent with all the existing splitting nodes
3. Split and recurse into subtrees

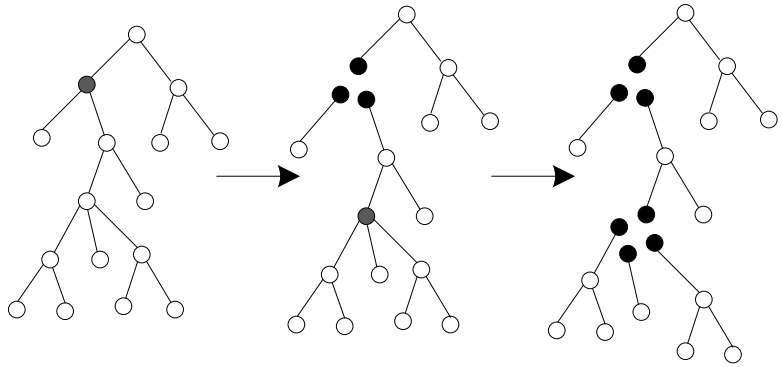
$\mathbf{SAT}_{\mathrm{tw}}(w(n))$ can be done in time $O^*(2^{w(n)\log n})$, space $O^*(1)$

Hybrid?

	Time-efficient		Space-efficient
Time	$O^*(2^{2w(n)})$	?	$O^*(2^{w(n)\log n})$
Space	$O^*(2^{w(n)})$	?	$O^*(1)$

Hybrid - First Attempt

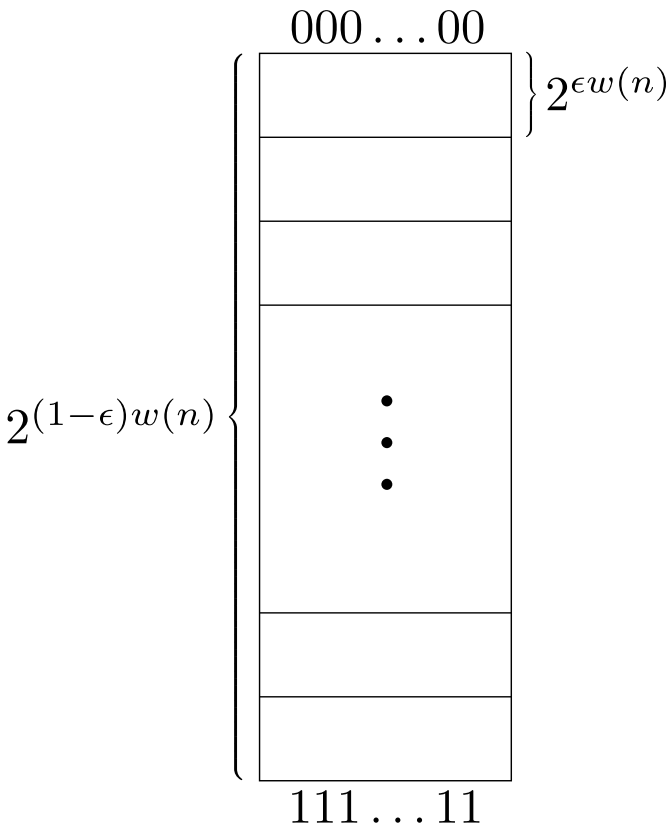


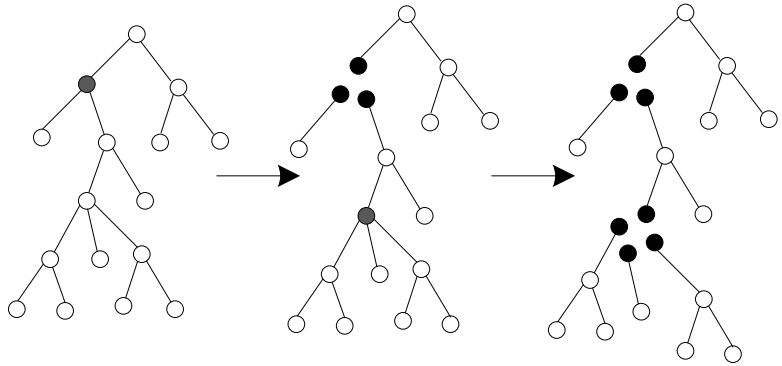


- Divide-and-conquer
- Memoization
  - a boolean value $\rightarrow$ an array

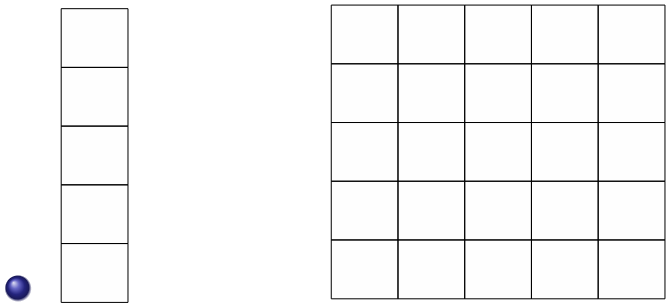
- up to $O^*(2^{\epsilon w(n) \log n})$ space
  - the worse of both algorithms

Hybrid - First Attempt



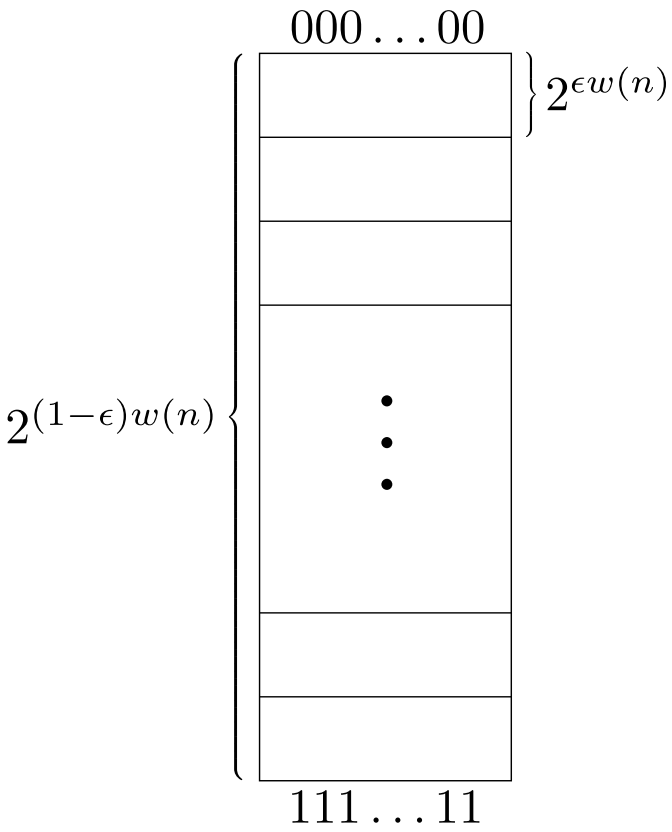


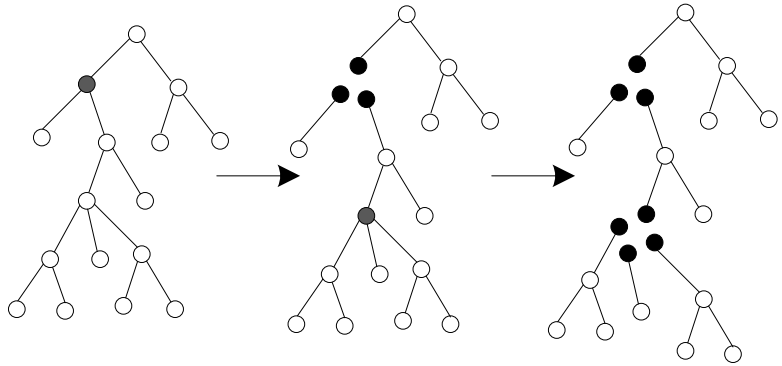
- Divide-and-conquer
- Memoization
  - a boolean value $\rightarrow$ an array



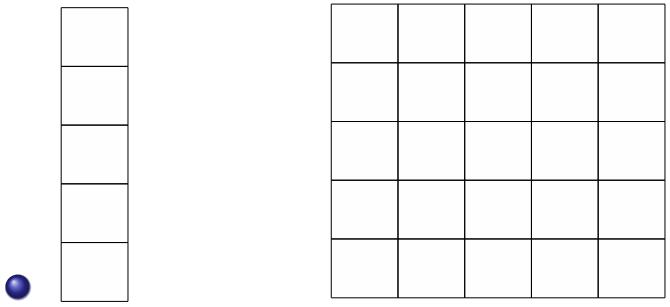
- up to $O^*(2^{\epsilon w(n) \log n})$ space
  - the worse of both algorithms

Hybrid - First Attempt



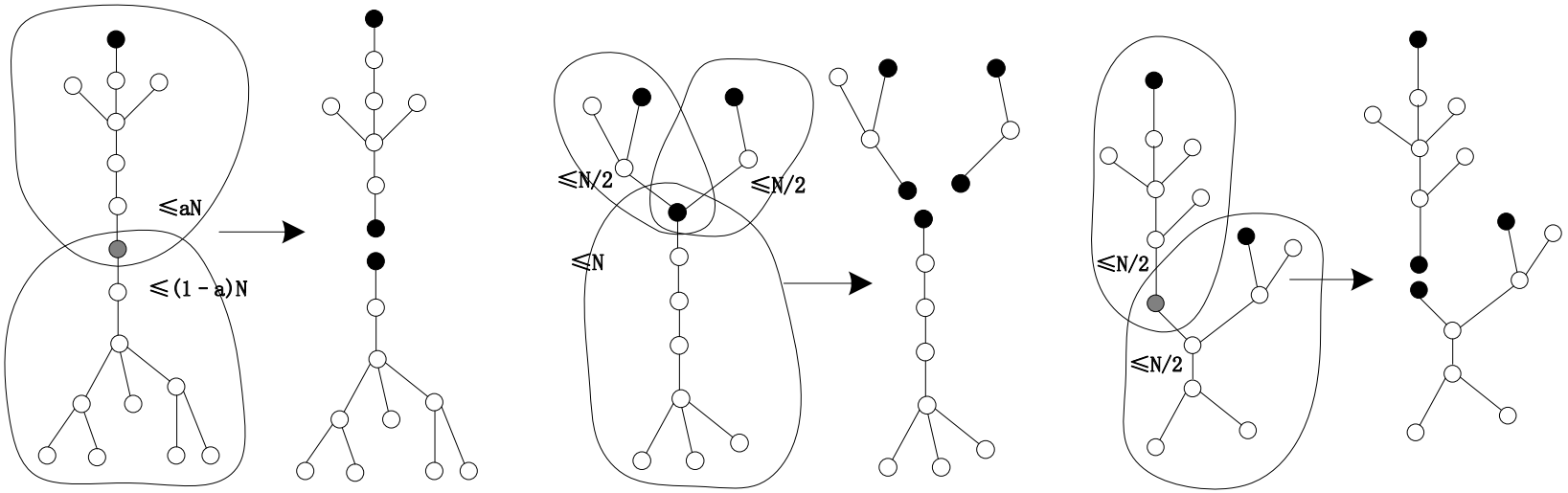


- Divide-and-conquer
- Memoization
  - a boolean value $\rightarrow$ an array



- up to $O^*(2^{\epsilon w(n) \log n})$ space
  - the worse of both algorithms

Splitting Strategy



- Idea: restrict # of active splitting nodes, ≤ 2
- space requirement $O^*(2^{2\epsilon w(n)})$

Hybrid

	Time-efficient	"Trade-off"	Space-efficient
Time	$O^*(2^{2w(n)})$	$O^*\left(2^{1.441(1-\epsilon)w(n)\log n}\right)$	$O^*(2^{w(n)\log n})$
Space	$O^*(2^{w(n)})$	$O^*\left(2^{2\epsilon w(n)}\right)$	$O^*(1)$

Conjecture

No such algorithm *asymptotically* in-between.

Hybrid

	Time-efficient	"Trade-off"	Space-efficient
Time	$O^*(2^{2w(n)})$	$O^*\left(2^{1.441(1-\epsilon)w(n)\log n}\right)$	$O^*(2^{w(n)\log n})$
Space	$O^*(2^{w(n)})$	$O^*\left(2^{2\epsilon w(n)}\right)$	$O^*(1)$

Conjecture

No such algorithm *asymptotically* in-between.

Assumptions on the Decomposition

1. # of nodes is $O(w(n)n)$ *nice* decompositions
2. The tree is of bounded degree *nice* decompositions
3. If a clause appears in one node, then all its variables also appear in the same node.
   - In [GP08], only for k-**SAT**, lost a constant factor in $\mathcal{TW}(\phi)$
   - Introduce clause bits into assignments

Assumptions on the Decomposition

1. # of nodes is $O(w(n)n)$ *nice* decompositions
2. The tree is of bounded degree *nice* decompositions
3. If a clause appears in one node, then all its variables also appear in the same node.
   - In [GP08], only for k-**SAT**, lost a constant factor in $\mathcal{TW}(\phi)$
   - Introduce clause bits into assignments

Assumptions on the Decomposition

1. # of nodes is $O(w(n)n)$ *nice* decompositions
2. The tree is of bounded degree *nice* decompositions
3. If a clause appears in one node, then all its variables also appear in the same node.
   - In [GP08], only for k-**SAT**, lost a constant factor in $\mathcal{TW}(\phi)$
   - Introduce clause bits into assignments

Assumptions on the Decomposition

1. # of nodes is $O(w(n)n)$ *nice* decompositions
2. The tree is of bounded degree *nice* decompositions
3. If a clause appears in one node, then all its variables also appear in the same node.
   - In [GP08], only for k-**SAT**, lost a constant factor in $\mathcal{TW}(\phi)$
   - Introduce clause bits into assignments

Main Results

Theorem

SAT$_{\text{tw}}(w(n))$ can be solved simultaneously in

- time $O^*\left(3^{(\lambda_c(\log n - c)+c)(1-\epsilon)w(n)}\right)$
- space $O^*\left(2^{c\epsilon w(n)}\right)$

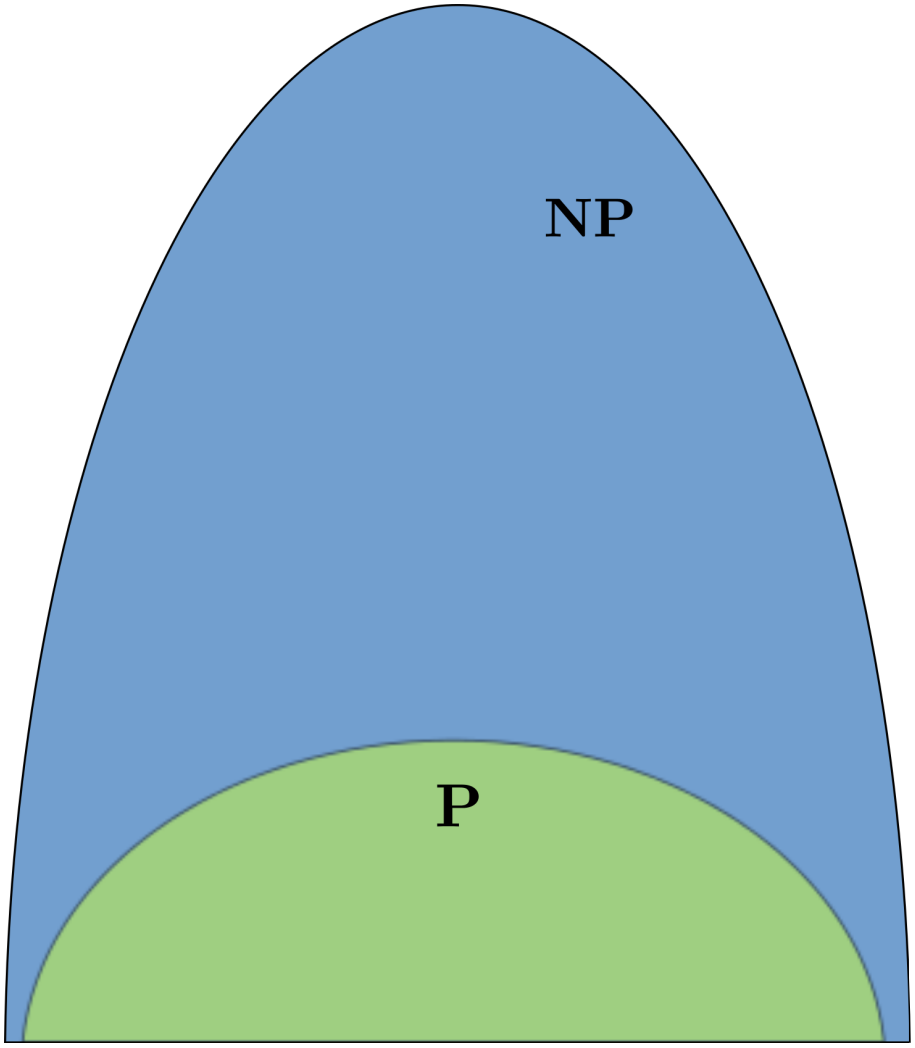
free parameters: ϵ, c

λ_c: $-\log X : X^c - X^{c-1} - X^{c-2} - \cdots - 1 = 0$, with largest $|X|$

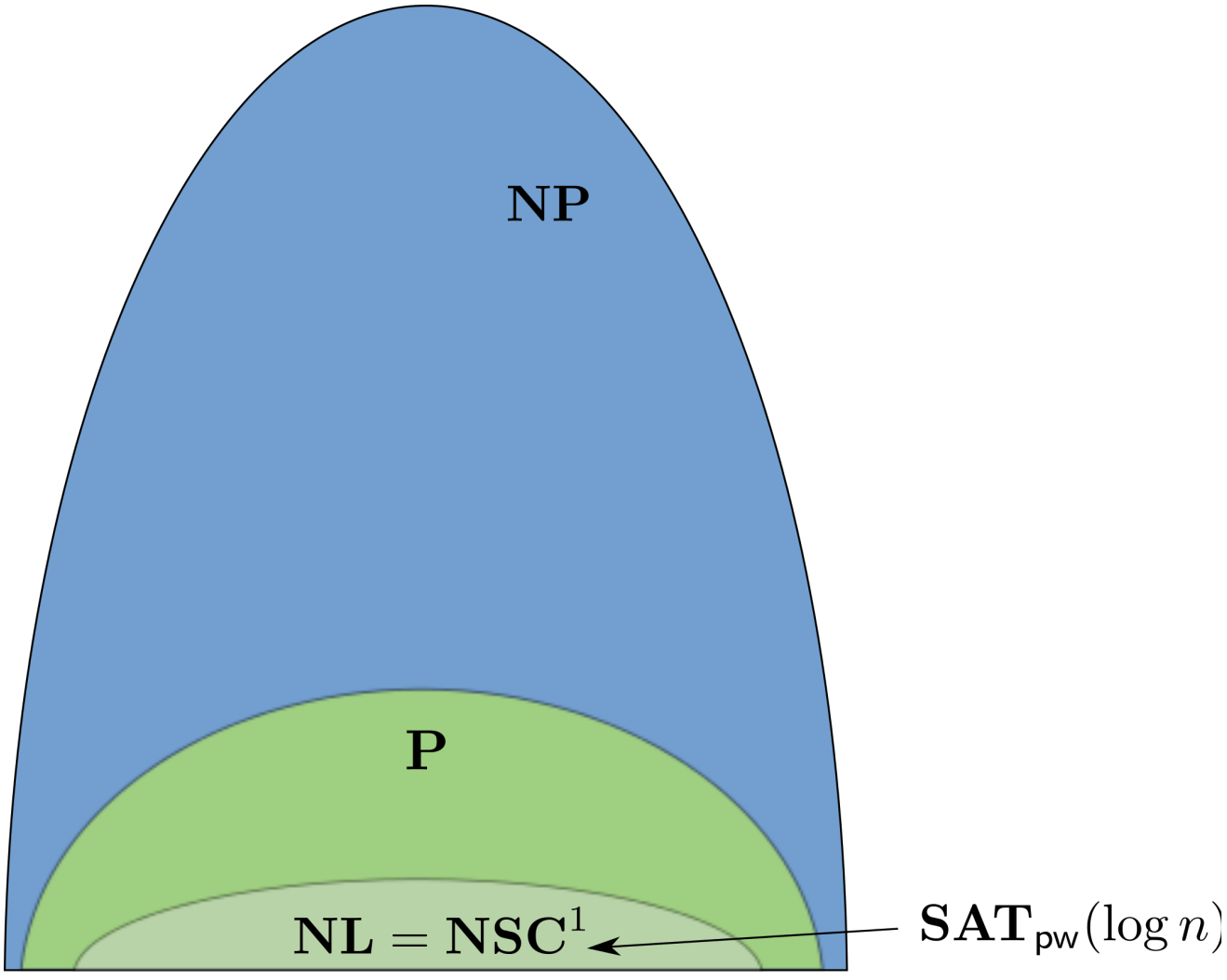
c	2	3	4	5	6	$\cdots$
λ_c	1.441	1.138	1.057	1.026	1.013	$\cdots$

Characterizations

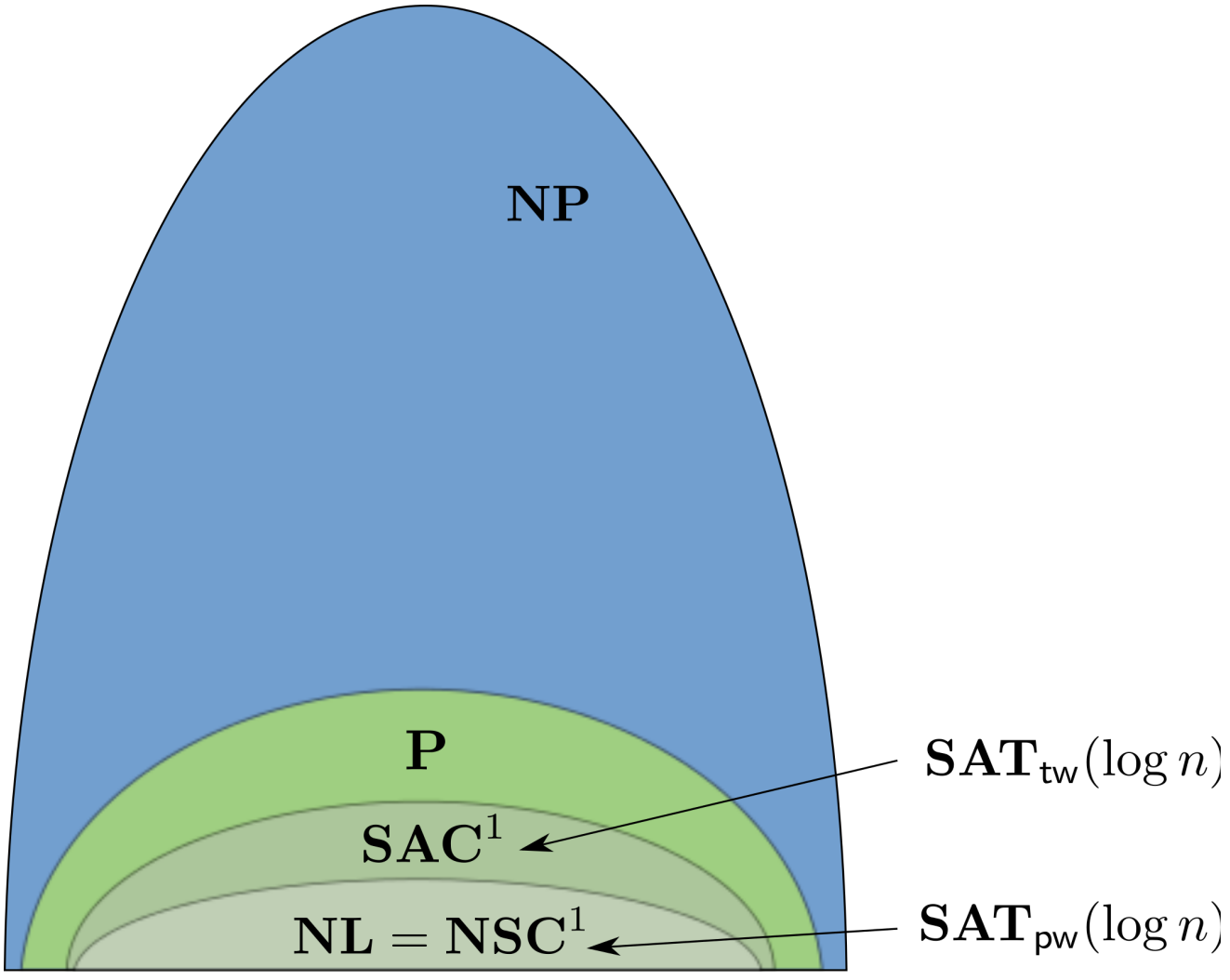
Characterizations



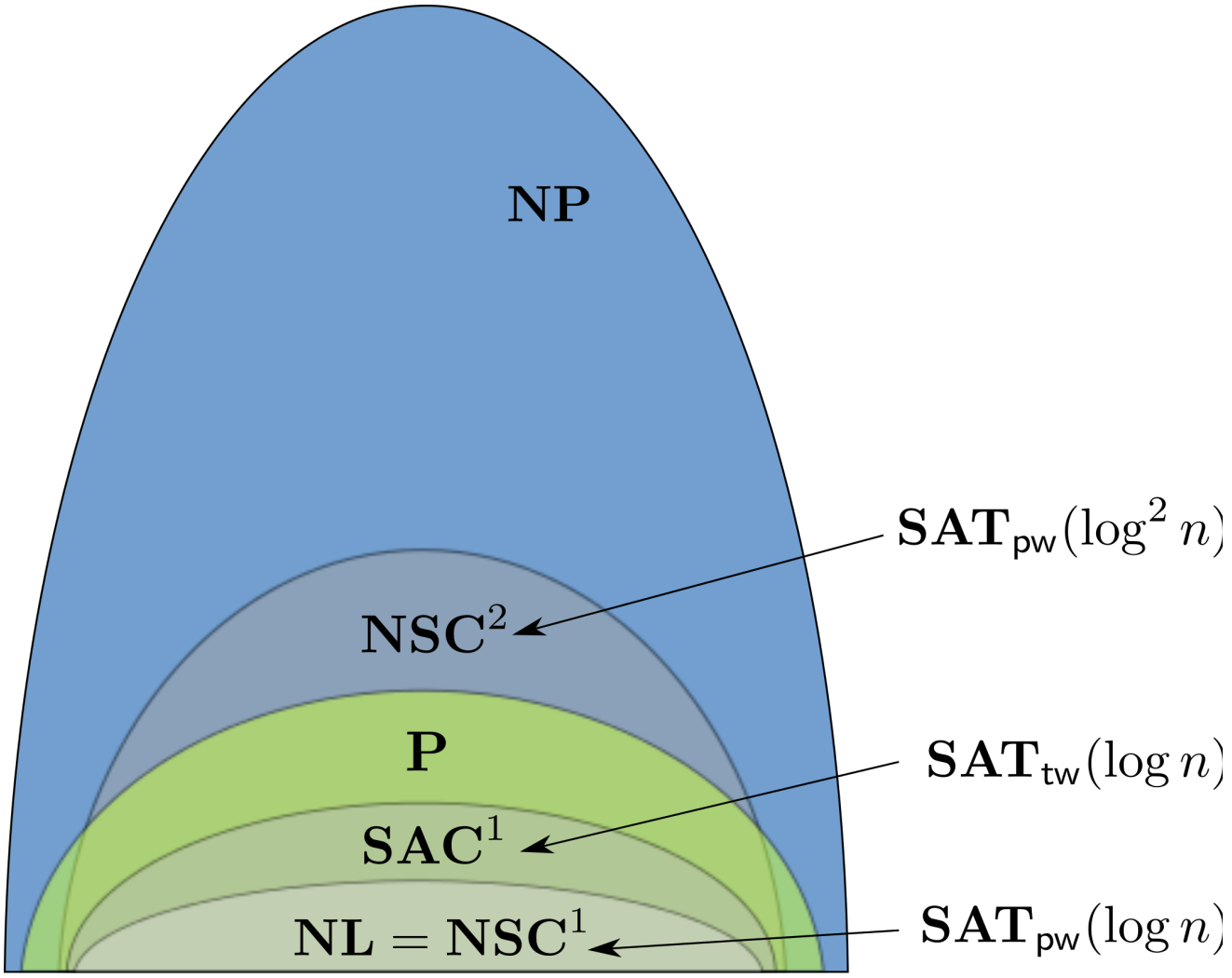
Characterizations



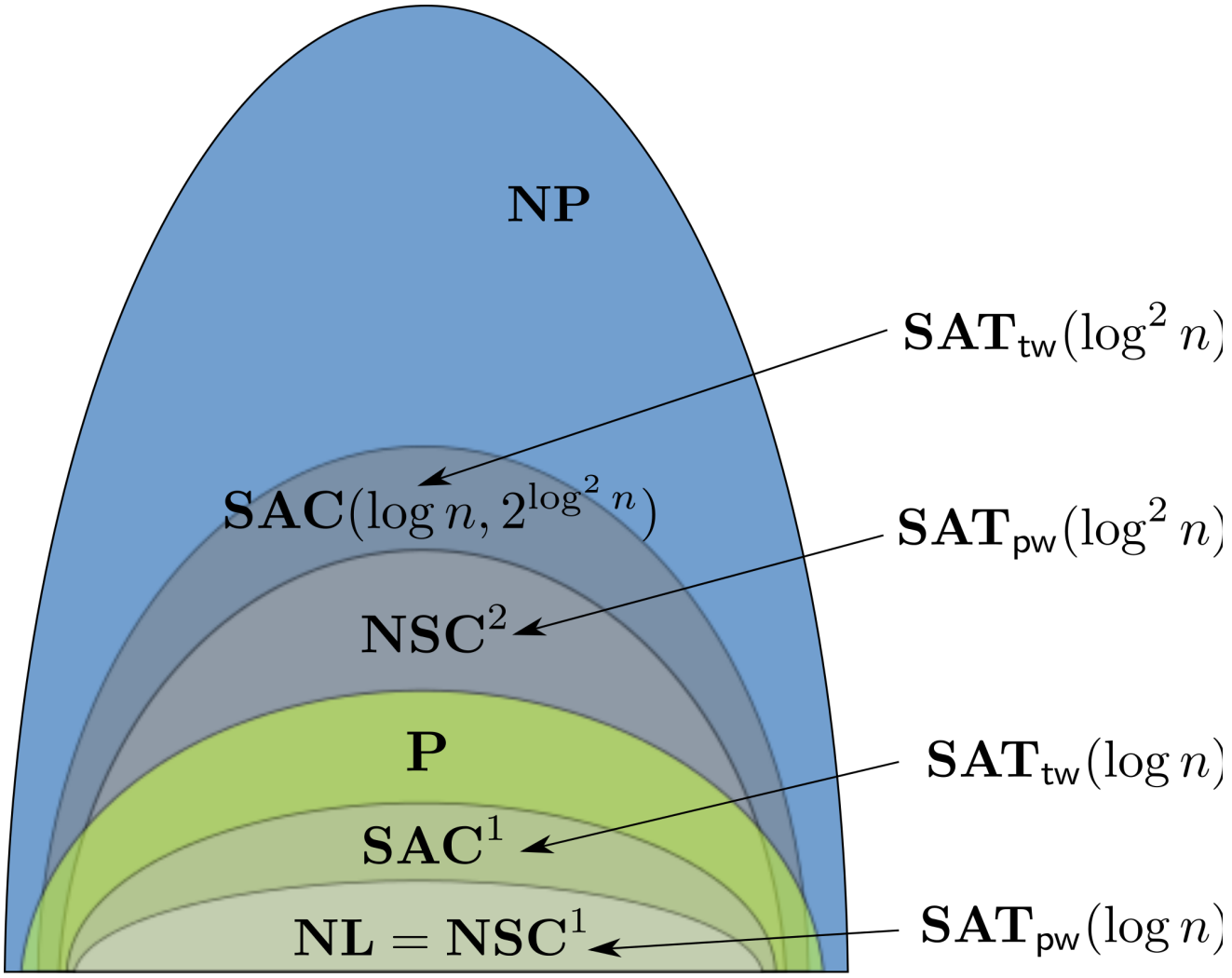
Characterizations



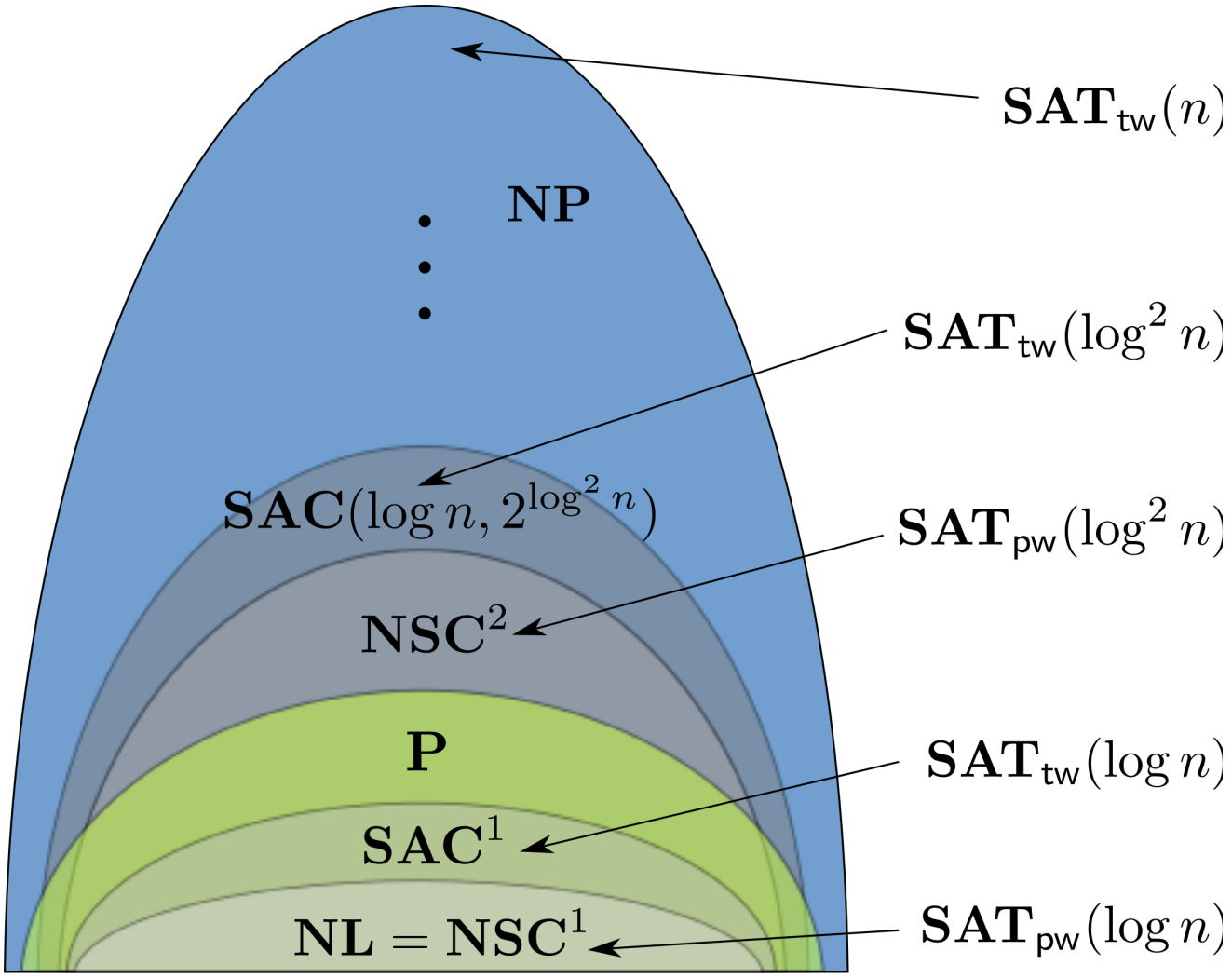
Characterizations



Characterizations



Characterizations



Our Conjecture

For $\mathcal{TW}(\phi) = \log^2 n$ or $\mathcal{PW}(\phi) = \log^2 n$,

$$\text{Conj for } \mathbf{SAT}_{\text{tw}} \Leftrightarrow \mathbf{SAC}(\log n, 2^{\log^2 n}) \not\subseteq \mathbf{TISP}(2^{\log^2 n}, n^{O(1)})$$
$$\Rightarrow \mathbf{SAC}^1 \not\subseteq \mathbf{SC}$$

$$\text{Conj for } \mathbf{SAT}_{\text{pw}} \Leftrightarrow \mathbf{NSC}^2 \not\subseteq \mathbf{TISP}(2^{\log^2 n}, n^{O(1)})$$
$$\Rightarrow \mathbf{NL} \not\subseteq \mathbf{SC}$$

Our Conjecture

For $\mathcal{TW}(\phi) = \log^2 n$ or $\mathcal{PW}(\phi) = \log^2 n$,

$$\text{Conj for } \mathbf{SAT}_{\text{tw}} \Leftrightarrow \mathbf{SAC}(\log n, 2^{\log^2 n}) \not\subseteq \mathbf{TISP}(2^{\log^2 n}, n^{O(1)})$$
$$\Rightarrow \mathbf{SAC}^1 \not\subseteq \mathbf{SC}$$

$$\text{Conj for } \mathbf{SAT}_{\text{pw}} \Leftrightarrow \mathbf{NSC}^2 \not\subseteq \mathbf{TISP}(2^{\log^2 n}, n^{O(1)})$$
$$\Rightarrow \mathbf{NL} \not\subseteq \mathbf{SC}$$

Summary

- Our Contributions
  - Algorithms for $\mathbf{SAT}_{tw}$
  - Complexity results and our conjecture
- Open Problems
  - Randomized algorithms?
  - Equivalence of our conjecture to well-know conjectures?