The SAT Problem
Structure through Visualization

Stephen Dunn

November 13, 2013

Some Terminology

- A $variable$ is a Boolean variable which can be assigned either T or F
- A $literal$ is a variable which may be negated ($\neg x$ vs x)

Some Terminology

- A *variable* is a Boolean variable which can be assigned either T or F
- A *literal* is a variable which may be negated ($\neg x$ vs x)
- We write OR as $\vee$, AND as $\wedge$
- Example:

  If we let $x =$ T, then $\neg x =$ F

- Example:

$$(A \vee \neg B \vee C) \wedge (A \vee B \vee \neg C) \wedge (\neg A \vee \neg B \vee C)$$

The SAT Problem

- Can we satisfy k literals in c clauses (CNF)?

The SAT Problem

- Can we satisfy k literals in c clauses (CNF)?
  - If yes, what are the assignments?
  - If no, exhaustive search needed? ($P \neq NP$)
- Problem is NP-complete (Cook '71)[1]
- Example:

$$(\neg A \vee B \vee \neg C) \wedge$$
$$(\neg A \vee \neg B \vee D) \wedge (\neg D)$$

The SAT Problem

- Can we satisfy k literals in c clauses (CNF)?
  - If yes, what are the assignments?
  - If no, exhaustive search needed? $(P \neq NP)$
- Problem is NP-complete (Cook '71)[1]
- Example:

$$(\neg A \vee B \vee \neg C) \wedge$$
$$(\neg A \vee \neg B \vee D) \wedge (\neg D)$$

Some solutions:

A=F, B=T, C=F, D=F
A=F, B=T, C=T, D=F
A=T, B=F, C=F, D=F

Satisfiable?

$$(A \vee B \vee C) \wedge$$
$$(\neg A \vee B \vee C) \wedge$$
$$(A \vee \neg B \vee C) \wedge$$
$$(\neg A \vee \neg B \vee C) \wedge$$
$$(A \vee B \vee \neg C) \wedge$$
$$(\neg A \vee B \vee \neg C) \wedge$$
$$(A \vee \neg B \vee \neg C) \wedge$$
$$(\neg A \vee B \vee D) \wedge$$
$$(\neg A \vee \neg B \vee \neg D)$$

Satisfiable?

$$(A \lor B \lor C) \land$$
$$(\neg A \lor B \lor C) \land$$
$$(A \lor \neg B \lor C) \land$$
$$(\neg A \lor \neg B \lor C) \land$$
$$(A \lor B \lor \neg C) \land$$
$$(\neg A \lor B \lor \neg C) \land$$
$$(A \lor \neg B \lor \neg C) \land$$
$$(\neg A \lor B \lor D) \land$$
$$(\neg A \lor \neg B \lor \neg D)$$

Yes! A = T, B = T, C = T, D = F

More Difficult?

$(A \vee B \vee C) \wedge$

$(\neg A \vee B \vee C) \wedge$

$(A \vee \neg B \vee C) \wedge$

$(\neg A \vee \neg B \vee C) \wedge$

$(A \vee B \vee \neg C) \wedge$

$(\neg A \vee B \vee \neg C) \wedge$

$(A \vee \neg B \vee \neg C) \wedge$

$(\neg A \vee \boldsymbol{\neg B} \vee D) \wedge$ $\leftarrow B$ is now negated

$(\neg A \vee \neg B \vee \neg D)$

More Difficult?

$(A \vee B \vee C) \wedge$

$(\neg A \vee B \vee C) \wedge$

$(A \vee \neg B \vee C) \wedge$

$(\neg A \vee \neg B \vee C) \wedge$

$(A \vee B \vee \neg C) \wedge$

$(\neg A \vee B \vee \neg C) \wedge$

$(A \vee \neg B \vee \neg C) \wedge$

$(\neg A \vee \boldsymbol{\neg B} \vee D) \wedge$ $\leftarrow B$ is now negated

$(\neg A \vee \neg B \vee \neg D)$

No solution!

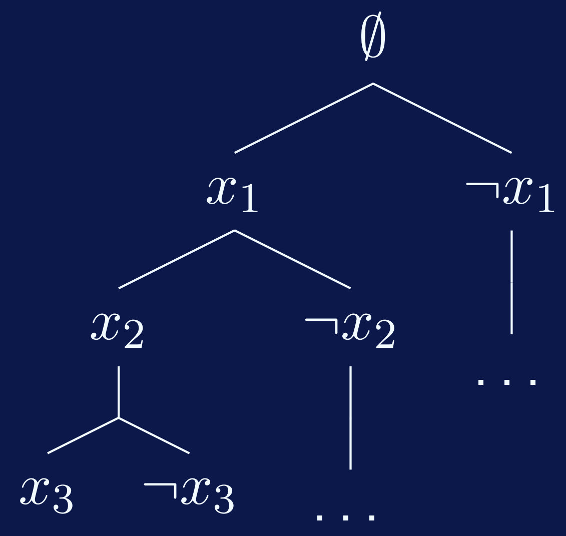
DP (DLL)

Algorithm 1 DP Algorithm[2] [3]

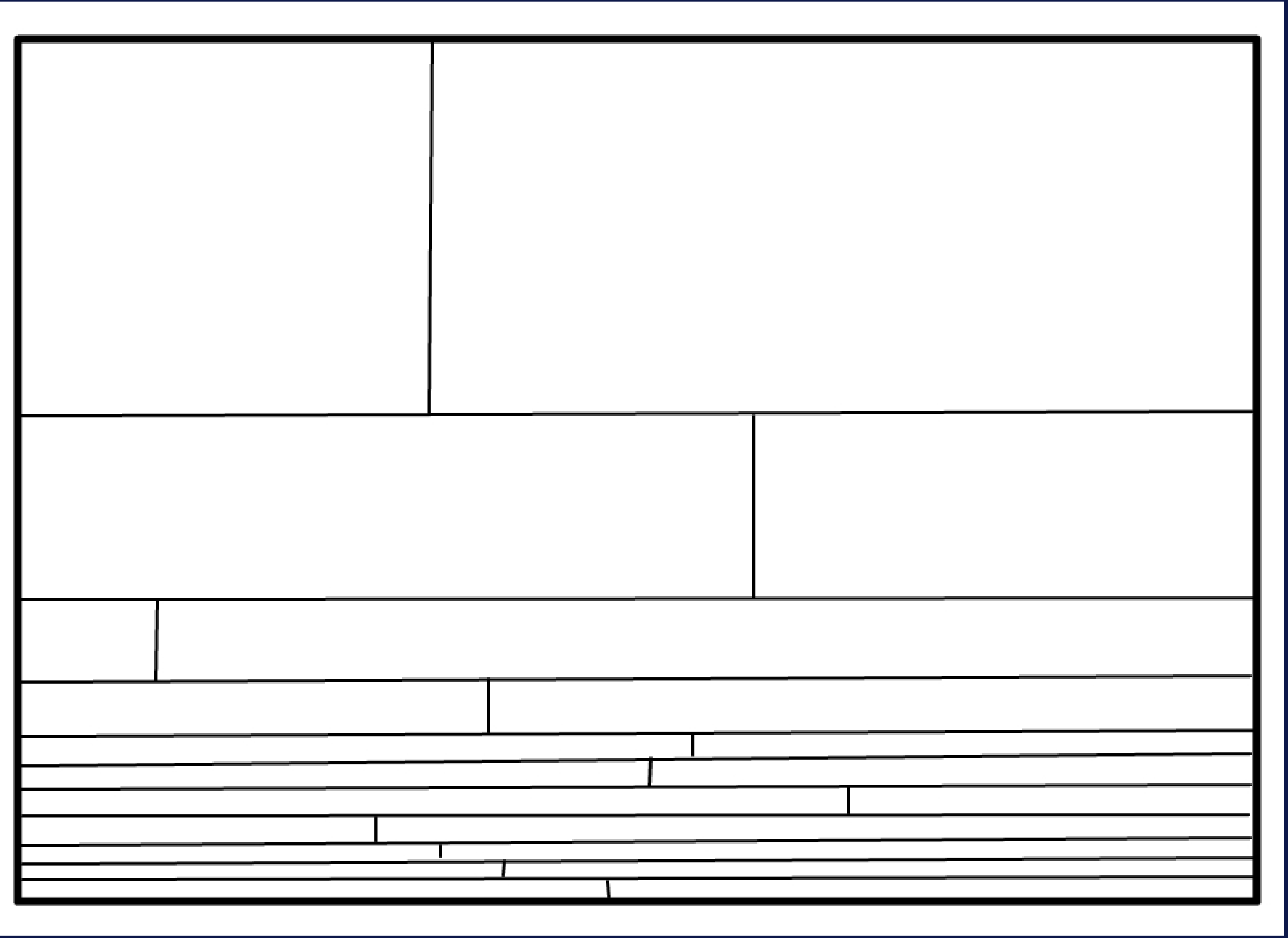
```
procedure DP(c)                                  ▷ Derive ∅ or F from CNF(c)
    while c ≠ {F} and c ≠ ∅ do
        δ ← SetAtomicLiterals(c)
        if |δ| ≤ 1 return c
        δ ← SetPureLiterals(c)
        if δ ≠ c then loop                       ▷ Propagate pure literals
        c ← SetVar(x ∈ δ, δ)
    end while
    return c
end procedure
```

Stochastic Search

- Randomly assign variables
- Improve working solution
  - Escape local minima
  - Avoid full state space
  - Determine unsat. without exhaustive search



State Space



Visualizations

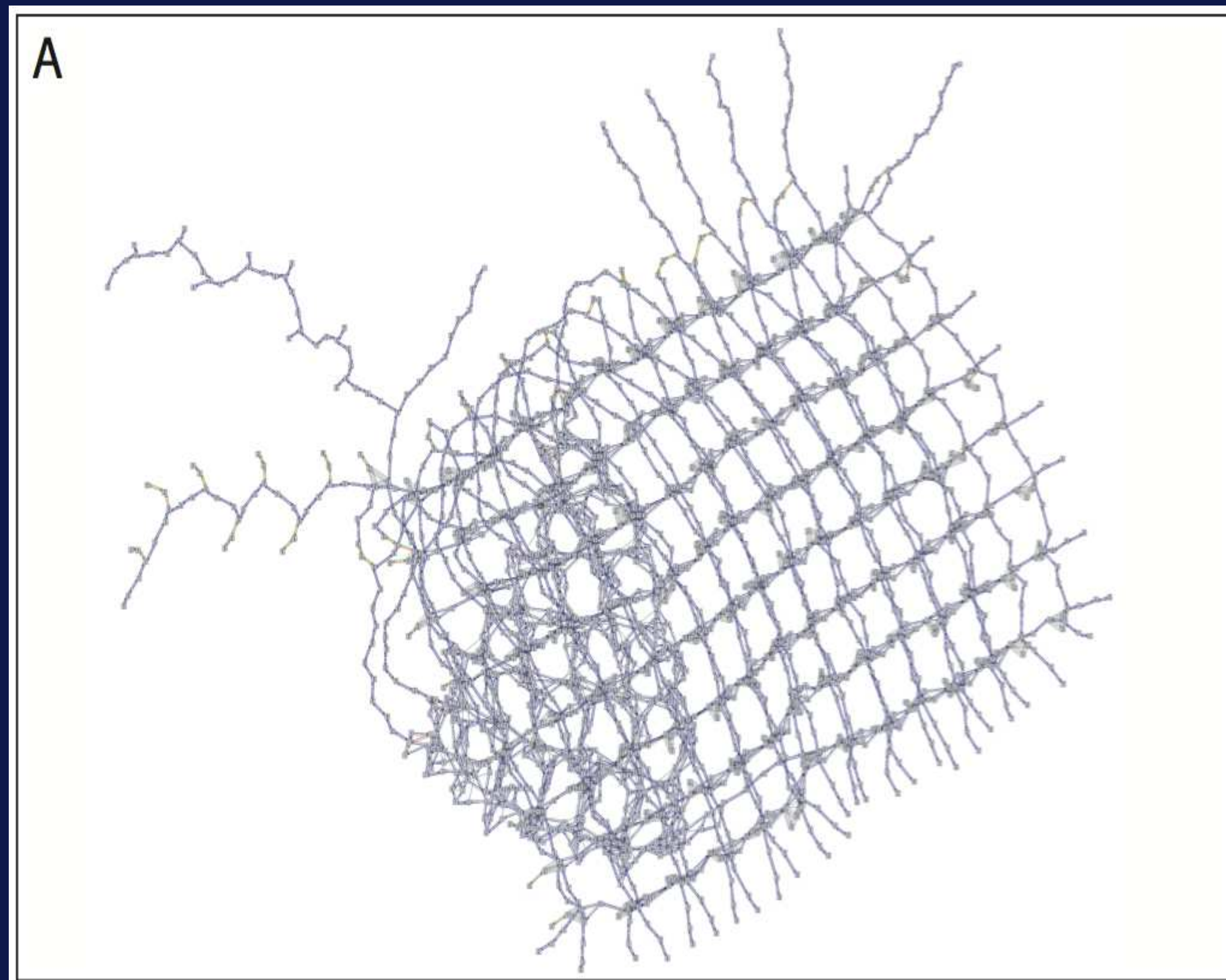


Figure 1: Literal-Based Renderings[4]

Visualizations (2)

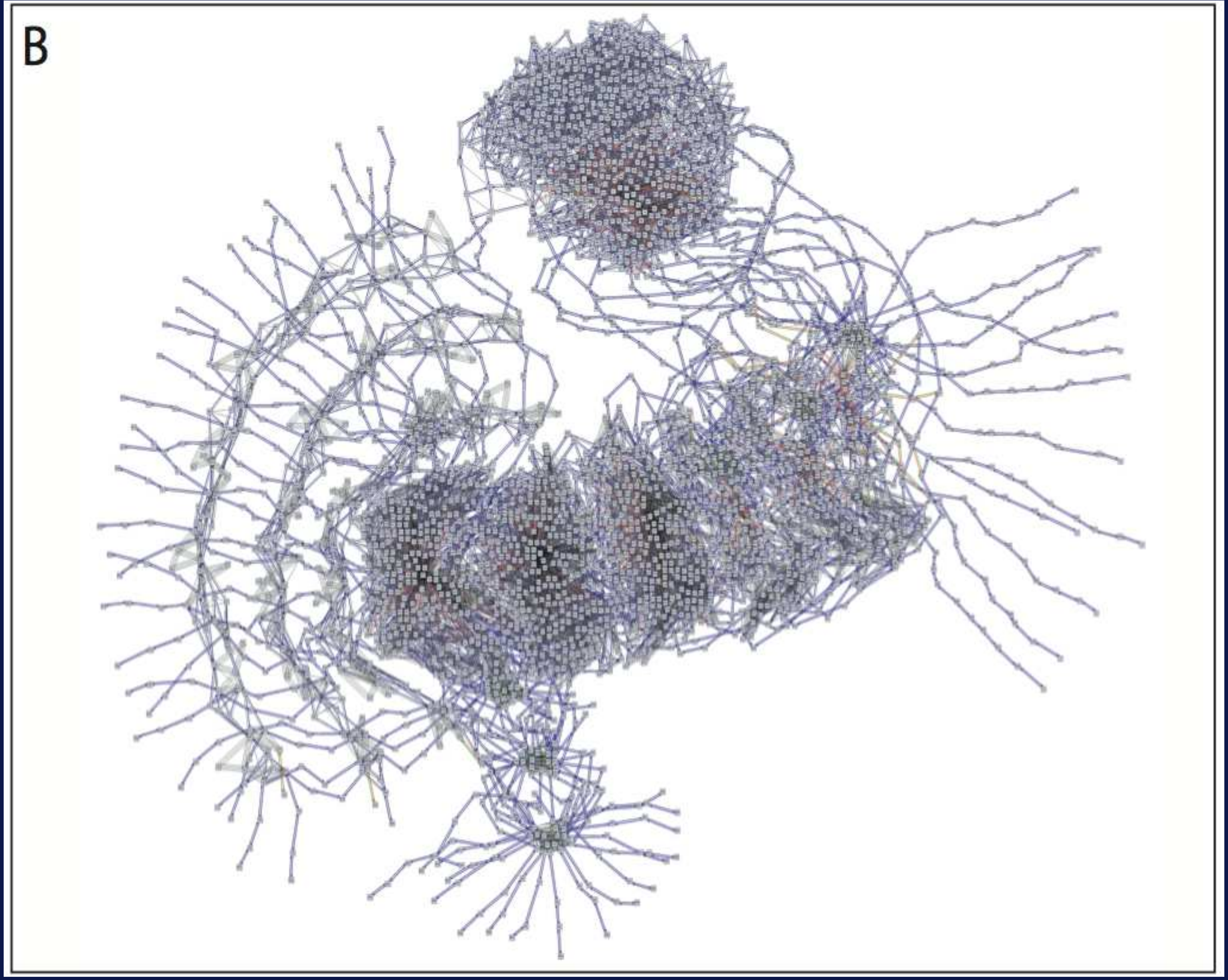


Figure 2: Literal-Based Renderings[4]

Visualizations (3)

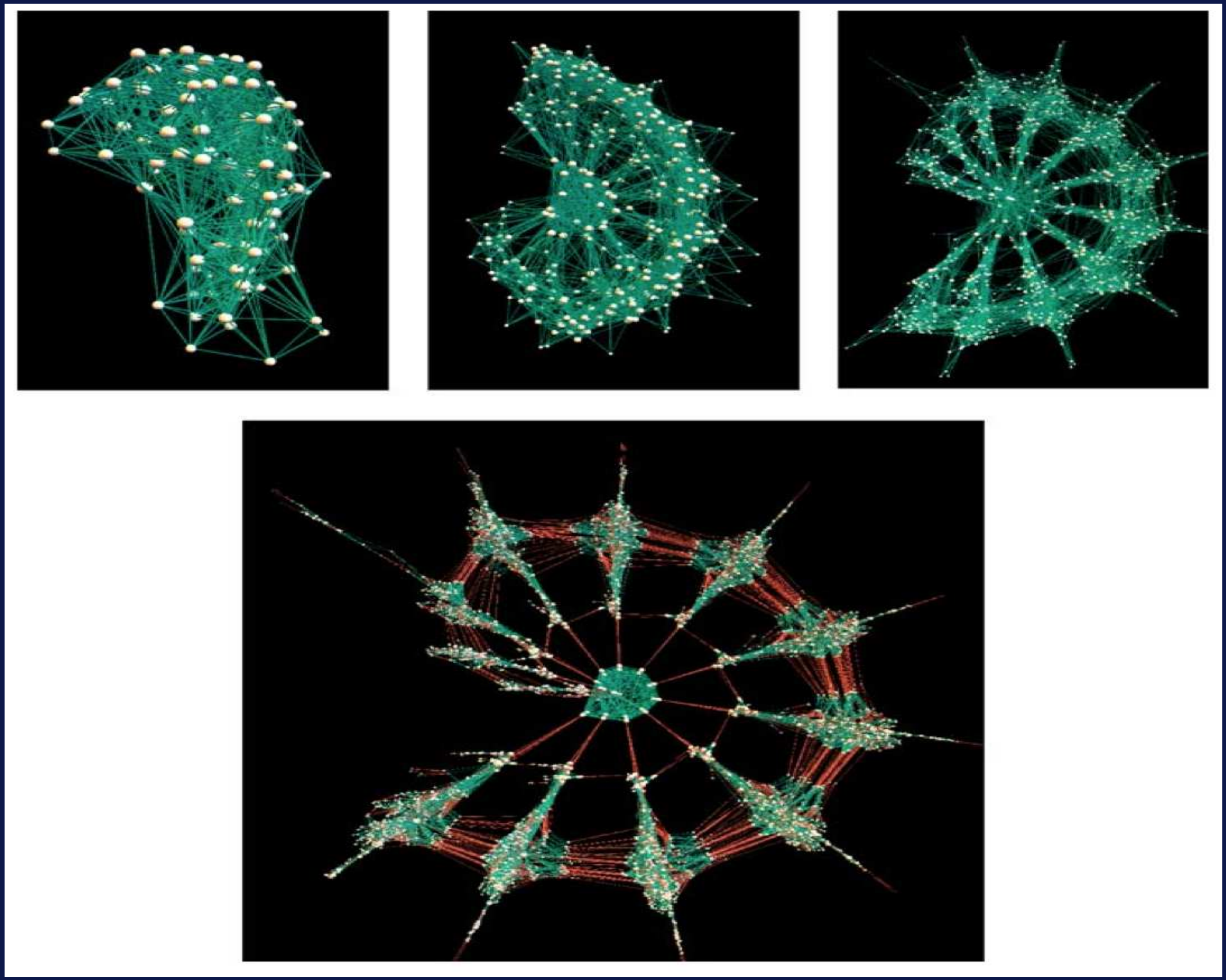


Figure 3: Clause-Based Renderings[5]

References

[1] Cook, S. A. (1971). **"The complexity of theorem-proving procedures"**. Proceedings of the 3rd Annual ACM Symposium on Theory of Computing: 151-158.

[2] Davis, Martin and Putnam, Hilary (1960). **"A Computing Procedure for Quantification Theory"**. Journal of the ACM 7 (3): 201-215.

[3] Davis, M.; Logemann, G.; Loveland, D. (1962). **"A machine program for theorem-proving"**. Communications of the ACM 5 (7): 394-397.

[4] Sinz, Carsten. **"Visualizing the internal structure of SAT instances (preliminary report)"**. SAT. Citeseer, 2004.

[5] Sinz, Carsten. **"Visualizing SAT instances and runs of the DPLL algorithm"**. Journal of Automated Reasoning (Vol. 39 Num. 2): 219-243. Springer, 2007.