DM865 – Spring 2018

Heuristics and Approximation Algorithms

Satisfiability

Marco Chiarandini

Department of Mathematics & Computer Science
University of Southern Denmark

Outline

1. SAT Problems

2. Dedicated Backtracking

3. Local Search for SAT

SAT Problem

Satisfiability problem in propositional logic

$$
\begin{aligned}
&(x_5 \vee x_8 \vee \bar{x}_2) \wedge (x_2 \vee \bar{x}_1 \vee \bar{x}_3) \wedge (\bar{x}_8 \vee \bar{x}_3 \vee \bar{x}_7) \wedge (\bar{x}_5 \vee x_3 \vee x_8) \wedge \\
&(\bar{x}_6 \vee \bar{x}_1 \vee \bar{x}_5) \wedge (x_8 \vee \bar{x}_9 \vee x_3) \wedge (x_2 \vee x_1 \vee x_3) \wedge (\bar{x}_1 \vee x_8 \vee x_4) \wedge \\
&(\bar{x}_9 \vee \bar{x}_6 \vee x_8) \wedge (x_8 \vee x_3 \vee \bar{x}_9) \wedge (x_9 \vee \bar{x}_3 \vee x_8) \wedge (x_6 \vee \bar{x}_9 \vee x_5) \wedge \\
&(x_2 \vee \bar{x}_3 \vee \bar{x}_8) \wedge (x_8 \vee \bar{x}_6 \vee \bar{x}_3) \wedge (x_8 \vee \bar{x}_3 \vee \bar{x}_1) \wedge (\bar{x}_8 \vee x_6 \vee \bar{x}_2) \wedge \\
&(x_7 \vee x_9 \vee \bar{x}_2) \wedge (x_8 \vee \bar{x}_9 \vee x_2) \wedge (\bar{x}_1 \vee \bar{x}_9 \vee x_4) \wedge (x_8 \vee x_1 \vee \bar{x}_2) \wedge \\
&(x_3 \vee \bar{x}_4 \vee \bar{x}_6) \wedge (\bar{x}_1 \vee \bar{x}_7 \vee x_5) \wedge (\bar{x}_7 \vee x_1 \vee x_6) \wedge (\bar{x}_5 \vee x_4 \vee \bar{x}_6) \wedge \\
&(\bar{x}_4 \vee x_9 \vee \bar{x}_8) \wedge (x_2 \vee x_9 \vee x_1) \wedge (x_5 \vee \bar{x}_7 \vee x_1) \wedge (\bar{x}_7 \vee \bar{x}_9 \vee \bar{x}_6) \wedge \\
&(x_2 \vee x_5 \vee x_4) \wedge (x_8 \vee \bar{x}_4 \vee x_5) \wedge (x_5 \vee x_9 \vee x_3) \wedge (\bar{x}_5 \vee \bar{x}_7 \vee x_9) \wedge \\
&(x_2 \vee \bar{x}_8 \vee x_1) \wedge (\bar{x}_7 \vee x_1 \vee x_5) \wedge (x_1 \vee x_4 \vee x_3) \wedge (x_1 \vee \bar{x}_9 \vee \bar{x}_4) \wedge \\
&(x_3 \vee x_5 \vee x_6) \wedge (\bar{x}_6 \vee x_3 \vee \bar{x}_9) \wedge (\bar{x}_7 \vee x_5 \vee x_9) \wedge (x_7 \vee \bar{x}_5 \vee \bar{x}_2) \wedge \\
&(x_4 \vee x_7 \vee x_3) \wedge (x_4 \vee \bar{x}_9 \vee \bar{x}_7) \wedge (x_5 \vee \bar{x}_1 \vee x_7) \wedge (x_5 \vee \bar{x}_1 \vee x_7) \wedge \\
&(x_6 \vee x_7 \vee \bar{x}_3) \wedge (\bar{x}_8 \vee \bar{x}_6 \vee \bar{x}_7) \wedge (x_6 \vee x_2 \vee x_3) \wedge (\bar{x}_8 \vee x_2 \vee x_5)
\end{aligned}
$$

Does there exist a truth assignment satisfying all clauses?

Search for a satisfying assignment (or prove none exists)

SAT Problem

Satisfiability problem in propositional logic

$$
\begin{aligned}
&(x_5 \vee x_8 \vee \bar{x}_2) \wedge (x_2 \vee \bar{x}_1 \vee \bar{x}_3) \wedge (\bar{x}_8 \vee \bar{x}_3 \vee \bar{x}_7) \wedge (\bar{x}_5 \vee x_3 \vee x_8) \wedge \\
&(\bar{x}_6 \vee \bar{x}_1 \vee \bar{x}_5) \wedge (x_8 \vee \bar{x}_9 \vee x_3) \wedge (x_2 \vee x_1 \vee x_3) \wedge (\bar{x}_1 \vee x_8 \vee x_4) \wedge \\
&(\bar{x}_9 \vee \bar{x}_6 \vee x_8) \wedge (x_8 \vee x_3 \vee \bar{x}_9) \wedge (x_9 \vee \bar{x}_3 \vee x_8) \wedge (x_6 \vee \bar{x}_9 \vee x_5) \wedge \\
&(x_2 \vee \bar{x}_3 \vee \bar{x}_8) \wedge (x_8 \vee \bar{x}_6 \vee \bar{x}_3) \wedge (x_8 \vee \bar{x}_3 \vee \bar{x}_1) \wedge (\bar{x}_8 \vee x_6 \vee \bar{x}_2) \wedge \\
&(x_7 \vee x_9 \vee \bar{x}_2) \wedge (x_8 \vee \bar{x}_9 \vee x_2) \wedge (\bar{x}_1 \vee \bar{x}_9 \vee x_4) \wedge (x_8 \vee x_1 \vee \bar{x}_2) \wedge \\
&(x_3 \vee \bar{x}_4 \vee \bar{x}_6) \wedge (\bar{x}_1 \vee \bar{x}_7 \vee x_5) \wedge (\bar{x}_7 \vee x_1 \vee x_6) \wedge (\bar{x}_5 \vee x_4 \vee \bar{x}_6) \wedge \\
&(\bar{x}_4 \vee x_9 \vee \bar{x}_8) \wedge (x_2 \vee x_9 \vee x_1) \wedge (x_5 \vee \bar{x}_7 \vee x_1) \wedge (\bar{x}_7 \vee \bar{x}_9 \vee \bar{x}_6) \wedge \\
&(x_2 \vee x_5 \vee x_4) \wedge (x_8 \vee \bar{x}_4 \vee x_5) \wedge (x_5 \vee x_9 \vee x_3) \wedge (\bar{x}_5 \vee \bar{x}_7 \vee x_9) \wedge \\
&(x_2 \vee \bar{x}_8 \vee x_1) \wedge (\bar{x}_7 \vee x_1 \vee x_5) \wedge (x_1 \vee x_4 \vee x_3) \wedge (x_1 \vee \bar{x}_9 \vee \bar{x}_4) \wedge \\
&(x_3 \vee x_5 \vee x_6) \wedge (\bar{x}_6 \vee x_3 \vee \bar{x}_9) \wedge (\bar{x}_7 \vee x_5 \vee x_9) \wedge (x_7 \vee \bar{x}_5 \vee \bar{x}_2) \wedge \\
&(x_4 \vee x_7 \vee x_3) \wedge (x_4 \vee \bar{x}_9 \vee \bar{x}_7) \wedge (x_5 \vee \bar{x}_1 \vee x_7) \wedge (x_5 \vee \bar{x}_1 \vee x_7) \wedge \\
&(x_6 \vee x_7 \vee \bar{x}_3) \wedge (\bar{x}_8 \vee \bar{x}_6 \vee \bar{x}_7) \wedge (x_6 \vee x_2 \vee x_3) \wedge (\bar{x}_8 \vee x_2 \vee x_5)
\end{aligned}
$$

Does there exist a truth assignment satisfying all clauses?

Search for a satisfying assignment (or prove none exists)

Motivation

- SAT used to solve many other problems!
- Applications:
  Hardware and Software Verification, Planning, Scheduling, Optimal Control, Protocol Design, Routing, Combinatorial problems, Equivalence Checking, etc.
- From 100 variables, 200 constraints (early 90s)
  to 1,000,000 vars. and 20,000,000 cls. in 20 years.

Propositional logic: Syntax

Propositional logic is the simplest logic—illustrates basic ideas
There are other types of logic: first-order logic, temporal logic, etc.

The proposition symbols x_1, x_2, etc. are sentences

If x is a sentence, $\neg x$ is a sentence (negation)
If x_1 and x_2 are sentences, $x_1 \wedge x_2$ is a sentence (conjunction)
If x_1 and x_2 are sentences, $x_1 \vee x_2$ is a sentence (disjunction)
If x_1 and x_2 are sentences, $x_1 \rightarrow x_2$ is a sentence (implication)
If x_1 and x_2 are sentences, $x_1 \leftrightarrow x_2$ is a sentence (biconditional)

Propositional logic: Semantics

Each model specifies true/false for each proposition symbol

E.g.	x_1	x_2	x_3
	true	*true*	*false*

(With these symbols, 8 possible models, can be enumerated automatically.)

Rules for evaluating truth with respect to a model m:

$\neg x$	is true iff	x	is false		
$x_1 \wedge x_2$	is true iff	x_1	is true *and*	x_2	is true
$x_1 \vee x_2$	is true iff	x_1	is true *or*	x_2	is true
$x_1 \rightarrow x_2$	is true iff	x_1	is false *or*	x_2	is true
i.e.,	is false iff	x_1	is true *and*	x_2	is false
$x_1 \leftrightarrow x_2$	is true iff	$x_1 \rightarrow x_2$	is true *and*	$x_2 \rightarrow x_1$	is true

Simple recursive process evaluates an arbitrary sentence, e.g.,

$\neg x_1 \wedge (x_2 \vee x_3) = \mathit{true} \wedge (\mathit{false} \vee \mathit{true}) \Leftrightarrow \mathit{true} \wedge \mathit{true} \Leftrightarrow \mathit{true}$

Truth tables for connectives

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \rightarrow Q$	$P \leftrightarrow Q$
false	*false*	*true*	*false*	*false*	*true*	*true*
false	*true*	*true*	*false*	*true*	*true*	*false*
true	*false*	*false*	*false*	*true*	*false*	*false*
true	*true*	*false*	*true*	*true*	*true*	*true*

Logical equivalence

Two sentences are logically equivalent iff true in same models:
$\alpha \equiv \beta$ if and only if $\alpha \models \beta$ and $\beta \models \alpha$

$$
\begin{array}{rcl}
(\alpha \wedge \beta) & \equiv & (\beta \wedge \alpha) \quad \text{commutativity of } \wedge \\
(\alpha \vee \beta) & \equiv & (\beta \vee \alpha) \quad \text{commutativity of } \vee \\
((\alpha \wedge \beta) \wedge \gamma) & \equiv & (\alpha \wedge (\beta \wedge \gamma)) \quad \text{associativity of } \wedge \\
((\alpha \vee \beta) \vee \gamma) & \equiv & (\alpha \vee (\beta \vee \gamma)) \quad \text{associativity of } \vee \\
\neg(\neg\alpha) & \equiv & \alpha \quad \text{double-negation elimination} \\
(\alpha \rightarrow \beta) & \equiv & (\neg\beta \rightarrow \neg\alpha) \quad \text{contraposition} \\
(\alpha \rightarrow \beta) & \equiv & (\neg\alpha \vee \beta) \quad \text{implication elimination} \\
(\alpha \leftrightarrow \beta) & \equiv & ((\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)) \quad \text{bicond. elimination} \\
\neg(\alpha \wedge \beta) & \equiv & (\neg\alpha \vee \neg\beta) \quad \text{De Morgan} \\
\neg(\alpha \vee \beta) & \equiv & (\neg\alpha \wedge \neg\beta) \quad \text{De Morgan} \\
(\alpha \wedge (\beta \vee \gamma)) & \equiv & ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) \quad \text{distributivity of } \wedge \text{ over } \vee \\
(\alpha \vee (\beta \wedge \gamma)) & \equiv & ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) \quad \text{distributivity of } \vee \text{ over } \wedge
\end{array}
$$

Validity and Satisfiability

A sentence is valid if it is true in all models,
e.g., $True$, $A \lor \neg A$, $A \to A$, $(A \land (A \to B)) \to B$

A sentence is satisfiable if it is true in some model
e.g., $A \lor B$, C

A sentence is unsatisfiable if it is true in no models
e.g., $A \land \neg A$

Conjunctive Normal Form

Every sentence in Propositional Logic is logically equivalent to a conjunction of clauses:

- A formula is in conjunctive normal form (CNF) iff it is of the form

$$\bigwedge_{i=1}^{m} \bigvee_{j=1}^{k_i} l_{ij} = (l_{11} \vee \ldots \vee l_{1k_1}) \wedge \ldots \wedge (l_{m1} \vee \ldots \vee l_{mk_m})$$

  where each literal l_{ij} is a propositional variable or its negation.

  The disjunctions of literlas: $c_i = (l_{i1} \vee \ldots \vee l_{ik_i})$ are called clauses.

- A formula is in k-CNF iff it is in CNF and all clauses contain exactly k literals (*i.e.*, for all i, $k_i = k$).
- In many cases, the restriction of SAT to CNF formulae is considered.
- For every propositional formula, there is an equivalent formula in 3-CNF.

Example:

$$
\begin{aligned}
F := \quad & \wedge (\neg x_2 \vee x_1) \\
& \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \\
& \wedge (x_1 \vee x_2) \\
& \wedge (\neg x_4 \vee x_3) \\
& \wedge (\neg x_5 \vee x_3)
\end{aligned}
$$

- F is in CNF.
- Is F satisfiable?
  Yes, *e.g.*, $x_1 := x_2 := \top$, $x_3 := x_4 := x_5 := \bot$ is a model of F.

Conversion to CNF

$x_1 \leftrightarrow (x_2 \vee x_3)$

1. Eliminate $\leftrightarrow$, replacing $\alpha \leftrightarrow \beta$ with $(\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)$.

   $(x_1 \rightarrow (x_2 \vee x_3)) \wedge ((x_2 \vee x_3) \rightarrow x_1)$

2. Eliminate $\rightarrow$, replacing $\alpha \rightarrow \beta$ with $\neg\alpha \vee \beta$.

   $(\neg x_1 \vee x_2 \vee x_3) \wedge (\neg(x_2 \vee x_3) \vee x_1)$

3. Move $\neg$ inwards using de Morgan's rules and double-negation:

   $(\neg x_1 \vee x_2 \vee x_3) \wedge ((\neg x_2 \wedge \neg x_3) \vee x_1)$

4. Apply distributivity law ($\vee$ over $\wedge$) and flatten:

   $(\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_2 \vee x_1) \wedge (\neg x_3 \vee x_1)$

SAT Problem

SAT Problem (decision problem, search variant):

- **Given:** Formula F in propositional logic
- **Task:** Find an assignment of truth values to variables in F that renders F true, or decide that no such assignment exists.

SAT Problem: A simple instance

- **Given:** Formula $F := (x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2)$
- **Task:** Find an assignment of truth values to variables x_1, x_2 that renders F true, or decide that no such assignment exists.

Special Cases

Not all instances are hard:

- Definite clauses: exactly one literal in the clause is positive. Eg:

  $\neg\beta \vee \neg\gamma \vee \alpha$

- Horn clauses: at most one literal is positive.

Easy interpretation: $\alpha \wedge \beta \rightarrow \gamma$ infers that $\neg\alpha \vee \neg\beta \vee \gamma$

Inference is easy by forward checking, linear time

Max SAT

Definition ((Maximum) K-Satisfiability (SAT))

Input: A set X of variables, a collection C of disjunctive clauses of at most k literals, where a literal is a variable or a negated variable in X.
k is a constant, $k > 2$.
Task: A truth assignment for X or a truth assignment that maximizes the number of clauses satisfied.

MAX-SAT (optimization problem)

Which is the maximal number of clauses satisfiable in a propositional logic formula F?

Outline

1. SAT Problems
2. Dedicated Backtracking
3. Local Search for SAT

DPLL algorithm

Davis, Putam, Logenmann & Loveland (DPLL) algorithm is a recursive depth-first enumeration of possible models with the following elements:

1. Early termination:
   a clause is true if any of its literals are true
   a formula is false if any of its clauses are false, which occurs when all its literals are false

2. Pure literal heuristic:
   pure literal is one that appears with same sign everywhere.
   it can be assigned so that it makes the clauses true. Clauses already true can be ignored.

3. Unit clause heuristic
   consider first unit clauses with just one literal or all literal but one already assigned. Generates cascade effect (forward chaining)

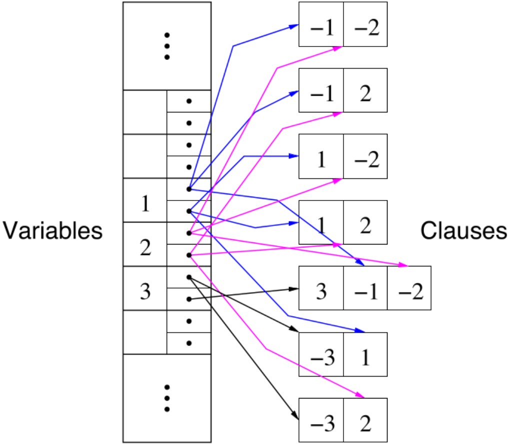
DPLL algorithm

Function DPLL(C, L, M)**:**

Data: C set of clauses; L set of literals; M model;
Result: *true* or *false*
if every clause in C is true in M **then return** *true*;
if some clause in C is false in M **then return** *false*;
$(l, val) \leftarrow$FindPureLiteral(L, C, M);
if l is non-null **then return** DPLL($C, L \setminus l, M \cup \{l = val\}$);
$(l, val) \leftarrow$FindUnitClause(L, M);
if l is non-null **then return** DPLL($C, L \setminus l, M \cup \{l = val\}$);
$l \leftarrow$First(L); $R \leftarrow$Rest(L);
return DPLL($C, R, M \cup \{l = true\}$) or
DPLL($C, R, M \cup \{l = false\}$)

Speedups

- Component analysis to find separable problems
- Intelligent backtracking
- Random restarts
- Clever indexing (data structures)
- Variable value ordering



Variable selection heuristics

- Degree
- Based on the occurrences in the (reduced) formula
    - Maximal Occurrence in clauses of Minimal Size (MOMS, Jeroslow-Wang)
- Variable State Independent Decaying Sum (VSIDS)
    - original idea (zChaff): for each conflict, increase the score of involved variables by 1, half all scores each 256 conflicts [MoskewiczMZZM2001]
    - improvement (MiniSAT): for each conflict, increase the score of involved variables by δ and increase $\delta := 1.05\delta$ [EenSörensson2003]

Value selection heuristics

- Based on the occurrences in the (reduced) formula
    - examples: Jeroslow-Wang, Maximal Occurrence in clauses of Minimal Size (MOMS), look-aheads

Outline

1. SAT Problems
2. Dedicated Backtracking
3. Local Search for SAT

Pre-processing

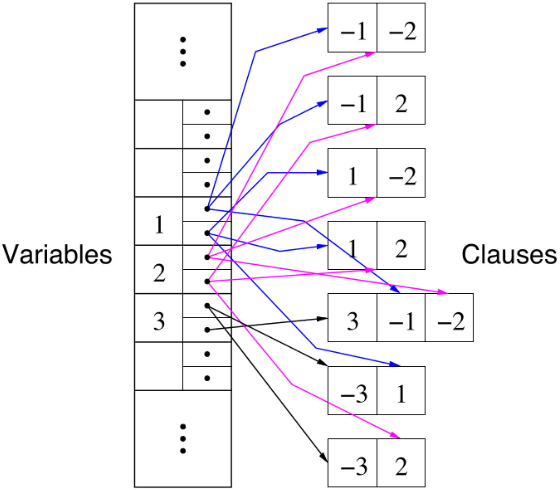
Pre-processing rules: low polynimial time procedures to decrease the size of the problem instance.

Typically applied in cascade until no rule is effective anymore.

Examples in SAT

1. eliminate duplicate literals
2. eliminate tautologies: $x_1 \vee \neg x_1 \ldots$
3. eliminate subsumed clauses
4. eliminate clauses with pure literals
5. eliminate unit clauses
6. unit propagation

Simple data structure for unit propagation



Maximum Weighted Satisfiability

Notation:

- 0-1 variables x_j, $j \in N = \{1, 2, \ldots, n\}$,
- clauses C_i, $i \in M = \{1, 2, \ldots, m\}$, and weights w_i (≥ 0), $i \in M$
- $\bar{x}_j = 1 - x_j$
- $L = \bigcup_{j \in N} \{x_j, \bar{x}_j\}$ set of literals
- $C_i \subseteq L$ for $i \in M$ (e.g., $C_i = \{x_1, \bar{x}_3, x_8\}$).
- Task: $\max_{\mathbf{x} \in \{0,1\}^n} \sum \{w_i \mid i \in M \text{ and } C_i \text{ is satisfied in } \mathbf{x}\}$

1. design one or more construction heuristics for the problem
2. devise preprocessing rules, ie, polynomial time simplification rules

Let's take the case $w_i = 1$ for all $i \in M$

- Assignment: $\mathbf{x} \in \{0, 1\}^n$
- Evaluation function: $f(\mathbf{x}) = \#$ unsatisfied clauses
- Neighborhood: one-flip
- Pivoting rule: best neighbor

Naive approach: exahustive neighborhood examination in $O(nmk)$ (k size of largest C_i)
A better approach:

- $C(x_j) = \{i \in M \mid x_j \in C_i\}$ (i.e., clauses dependent on x_j)
- $L(x_j) = \{\ell \in N \mid \exists i \in M \text{ with } x_\ell \in C_i \text{ and } x_j \in C_i\}$
- $f(\mathbf{x}) = \#$ unsatisfied clauses
- $\Delta(x_j) = f(\mathbf{x}) - f(\mathbf{x}'), \qquad \mathbf{x}' = \delta_{1E}^{x_j}(\mathbf{x})$ (aka score of x_j)

Initialize:

- compute f, score of each variable, and list unsat clauses in $O(mk)$
- init $C(x_j)$ for all variables

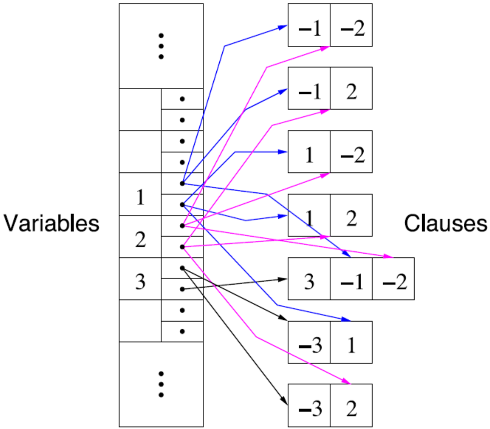
Examine Neighborhood

- choose the var with best score

Update:

- change the score of variables affected, that is, look in $C(\cdot)$ $O(mk)$

$C(x_j)$ Data Structure



Even better approach (though same asymptotic complexity):
$\rightsquigarrow$ after the flip of x_j only the score of variables in $L(x_j)$ that critically depend on x_j actually changes

- Clause C_i is critically satisfied by a variable x_j in $\mathbf{x}$ iff:
  - x_j is in C_i
  - C_i is satisfied in $\mathbf{x}$ and flipping x_j makes C_i unsatisfied (e.g., $1 \vee 0 \vee 0$ but not $1 \vee 1 \vee 0$)

  Keep a list of such clauses for each var
- x_j is critically dependent on x_ℓ under $\mathbf{x}$ iff:
  there exists $C_i \in C(x_j) \cap C(x_\ell)$ and such that flipping x_j:
  - C_i changes from satisfied to not satisfied or viceversa
  - C_i changes from satisfied to critically satisfied by x_ℓ or viceversa

Initialize:

- compute score of variables;
- init $C(x_j)$ for all variables
- init status criticality for each clause (ie, count # of ones per clause)

Update:

change sign to score of x_j
for all C_i in $C(x_j)$ where critically dependent vars are **do**
    for all $x_\ell \in C_i$ **do**
        update score x_ℓ depending on its critical status before flipping x_j

Summary

1. SAT Problems
2. Dedicated Backtracking
3. Local Search for SAT