Coping with NP-Hardness

- Strategies to deal with hard problems
- Algorithms for Special Cases
- Fixed Parameter Tractability
- Intelligent Exhaustive Search
  - Backtracking
  - Branch and Bound
- Dynamic Programming based pseudo polynomial algorithm TSP

Imdad ullah Khan

Coping with NP-Hardness

Approaches to tackle hard problems

1. Special Cases: Relevant structure on which the problem is easy
   - Exact results in poly-time only for special cases or a range of parameters
2. Intelligent Exhaustive Search: Exponential time in worst case
   - The base and/or exponent are usually smaller
   - Could be efficient on typical more realistic instances
   - Backtracking, Brand-and-Bound
3. Nearly exact solutions: Output is *'close'* to exact (optimal) solution
   - Approximation Algorithms: Solutions of guaranteed quality in poly-time
   - Heuristic: Solutions hopefully good in poly-time
4. Randomized Algorithms: Use coin flips for making decisions
   - Typically used for approximation, also used problems in P

Coping with NP-HARDNESS: Special Cases

- $3d$ − MATCHING is NP-HARD
  - The "2d" special case, is just graph matching problem
- VERTEX-COVER(G, k) is NP-HARD
  - When G is bipartite, it is the dual of BIPARTITE-MATCHING
- SAT(f) is NP-HARD
  - The special case of $2 -$ SAT(f) is easy
- WEIGHTED-INDEPENDENT-SET(G) is NP-HARD
  - When G is a path, it can be solved easily with dynamic programming
- INDEPENDENT-SET(G) is NP-HARD
  - When G is a tree, the problem can be solved in polynomial time

The 2-SAT Search Problem

- Given n Boolean variables $x_1, \ldots, x_n$ ▷ taking value of $0/1$
- A literal is a variable appearing in some formula as x_i or $\bar{x}_i$
- A clause of size 2 is an OR of two literals
- A 2-CNF formula is AND of one or more clauses of size ≤ 2
- A formula is satisfiable if there is an assignment of $0/1$ values to the variables such that the formula evaluates to 1 (or true)

2-SAT(f) search problem: Find a satisfying assignment for f if one exists?

$(x \vee y) \wedge (\overline{x} \vee z) \wedge (\overline{z})$ is satisfied with $x = 0,\ y = 1,\ z = 0$

$(x \vee y) \wedge (y) \wedge (\overline{x} \vee z) \wedge (\overline{y})$ is not satisfiable

The 2-SAT Search Problem

Meaning of a clause in 2-CNF formula

A clause (ℓ_1) means ℓ_1 must be true or it cannot be false

A clause $(\ell_1 \vee \ell_2)$ means one of ℓ_1 & ℓ_2 must be true (both can't be false)

- i.e. if $\ell_1 = 0$, then $\ell_2 = 1$ and if $\ell_2 = 0$, then $\ell_1 = 1$
- In other words, if $\overline{\ell_1} = 1$, then $\ell_2 = 1$ and if $\overline{\ell_2} = 1$, then $\ell_1 = 1$

A 2-cnf formula is a series of implications of the above form

Implications are transitive $\big[(a \to b) \text{ AND } (b \to c)\big] \longrightarrow (a \to c)$

From $(\overline{x} \vee y) \wedge (\overline{y} \vee z)$ we get implications

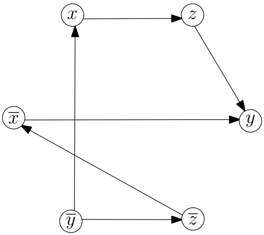
$$\big[(x = 1 \to y = 1) \text{ AND } (y = 1 \to z = 1)\big] \longrightarrow (x = 1 \to z = 1)$$

The 2-SAT Search Problem

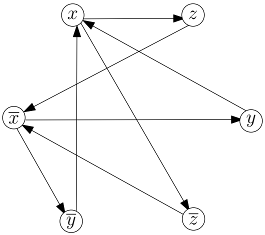
The Implication Graph for a 2-CNF formula f is a digraph $G = (V, E)$

- V are variables of f and their negations (all literals)
- E correspond to the two implications from each clause

$(\mathbf{x} \vee \mathbf{y}) \wedge (\overline{\mathbf{x}} \vee \mathbf{z}) \wedge (\mathbf{y} \vee \overline{\mathbf{z}})$



$(\mathbf{x} \vee \mathbf{y}) \wedge (\overline{\mathbf{x}} \vee \mathbf{z}) \wedge (\overline{\mathbf{x}} \vee \overline{\mathbf{z}}) \wedge (\mathbf{x} \vee \overline{\mathbf{y}})$



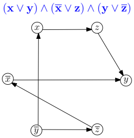
The left formula has a satisfying solution and the right one has none

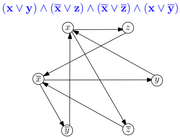
How is this fact depicted in the graph?

The 2-SAT Search Problem

The Implication Graph for a 2-CNF formula f is a digraph $G = (V, E)$

- V are variables of f and their negations (all literals)
- E correspond to the two implications from each clause



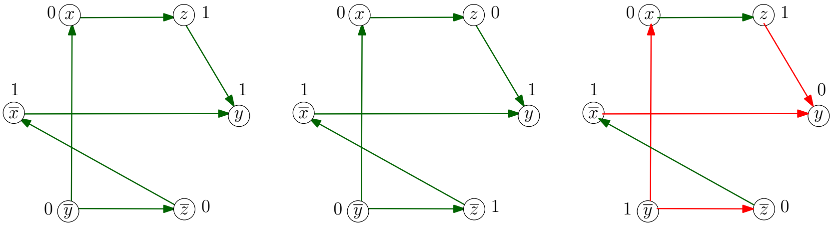


- All clauses are satisfied $\equiv$ all implications (now edges) are true
- An implication $x \to y$ is true always except for $x = 1$ and $y = 0$
- All edges satisfied, meaning there is no edge (x, y), with the vertex (literal) x has value 1 and the vertex (literal) y has value 0

We want an assignment to variables so there is no edge from 1 to 0

The 2-SAT Search Problem

$$(\mathbf{x} \vee \mathbf{y}) \wedge (\overline{\mathbf{x}} \vee \mathbf{z}) \wedge (\mathbf{y} \vee \overline{\mathbf{z}})$$



$x = 0, y = 1, z = 1$ $\qquad$ $x = 0, y = 1, z = 0$ $\qquad$ $x = 0, y = 0, z = 1$

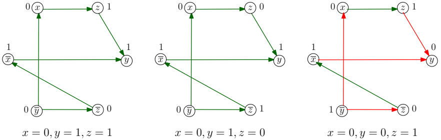
For the above formula (implication graph)

- $(x, y, z) = (0, 1, 1)$ satisfy all edges
- $(x, y, z) = (0, 1, 0)$ satisfy all edges
- $(x, y, z) = (0, 0, 1)$ does not satisfy the red edges

We try to satisfy all the edges of the corresponding implication graph

The 2-SAT Search Problem

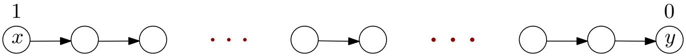
$(\mathbf{x} \vee \mathbf{y}) \wedge (\overline{\mathbf{x}} \vee \mathbf{z}) \wedge (\mathbf{y} \vee \overline{\mathbf{z}})$



1) Make an implication graph from the formula and 2) find an assignment to vertices that is not-conflicting ($\ell \neq \bar{\ell}$) and all edges are satisfied

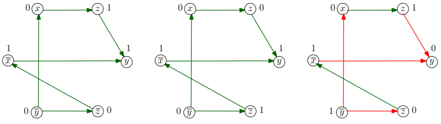
In any assignment that satisfies all edges, there cannot be a 1 to 0 edge

In any assignment that satisfies all edges, there cannot be a 1 to 0 path



The 2-SAT Search Problem

$(\mathbf{x} \vee \mathbf{y}) \wedge (\overline{\mathbf{x}} \vee \mathbf{z}) \wedge (\mathbf{y} \vee \overline{\mathbf{z}})$

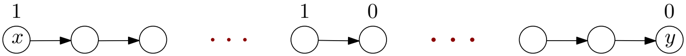


$x = 0, y = 1, z = 1$ $x = 0, y = 1, z = 0$ $x = 0, y = 0, z = 1$

1) Make an implication graph from the formula and 2) find an assignment to vertices that is not-conflicting ($\ell \neq \bar{\ell}$) and all edges are satisfied

In any assignment that satisfies all edges, there cannot be a 1 to 0 edge

In any assignment that satisfies all edges, there cannot be a 1 to 0 path



This is the transitive property of implications

The 2-SAT Search Problem

1) Make an implication graph from the formula and 2) find an assignment to vertices that is not-conflicting ($\ell \neq \bar{\ell}$) and all edges are satisfied

In any assignment that satisfies all edges, there cannot be a 1 to 0 path

- If there is a path from u to v, we should not make $u = 1$ and $v = 0$
  - ▷ Can we check all paths? what if there are bidirectional paths?
- Whenever there is a path from u to v and a path from v to u, then u and v must be assigned the same value
  - All literals lying in the same strongly connected components, must be assigned the same value
- If a literal and its negation are in the same strongly connected components, the formula is not satisfiable
  - ▷ Indeed, that is the only way a formula would not be satisfiable

The 2-SAT Search Problem

1) Make an implication graph from the formula and 2) find an assignment to vertices that is not-conflicting ($\ell \neq \bar{\ell}$) and all edges are satisfied

In any assignment that satisfies all edges, there cannot be a 1 to 0 path

1. Find strongly connected components of the implication graphs
2. Give each component the same value

   ▷ Need to make sure that there is no path from 1 to 0

   ▷ Which component should get 1 which should get 0?

   The component graph is a DAG
3. Traverse vertices in reverse topological ordering of their SCC's
4. If literals in current SCC are not assigned
   - Set all of them to 1   Set their negations to 0

The 2-SAT Search Problem

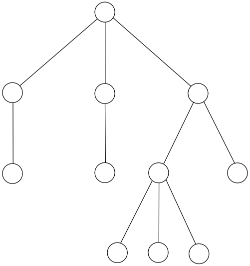
If no literal and its negation are in the same components, then the above algorithm produce a valid and satisfying assignment

- If a literal is set to 1, then all the literals reachable from it have already been set to 1, because we are processing literals in reverse topological order
- If a literal is set to 0, then all the literals reachable from it have already been set to 0, because the above statement is true about the corresponding edges (skew symmetry)

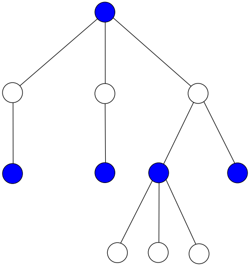
Max-Independent-Set in Trees

- max-independent-set(G) is np-hard (reduction form decision version)
- Can be solved efficiently when G is a tree or forest (acyclic)

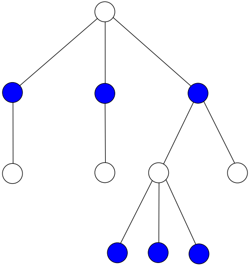
Max-Independent-Set in Trees



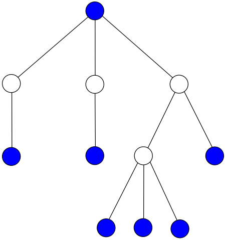
A tree on 11 vertices



An Independent set of size 5



An Independent set of size 6



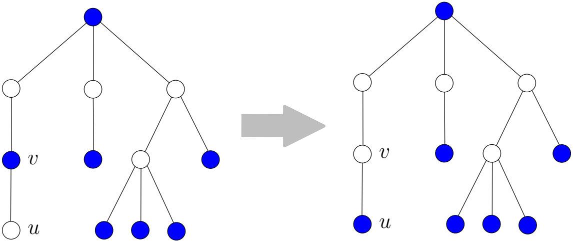
Max Independent set of size 7

Max-Independent-Set in Trees

Any tree has at least one leaf (actually two)

For any leaf u in T, there is a max independent set containing u

- Let S be a max independent set, $S \not\ni u$
- Let v be the only neighbor of u ▷ u has degree 1
- $v \in S$, otherwise u can be in S contradicting maximality of S
- Exchange u for v



Max-Independent-Set in Trees

Any tree has at least one leaf (actually two)

For any leaf u in T, there is a max independent set containing u

For any leaf u in T, a max independent set is $\{u\}$ union a max independent set in $T \setminus \{u\}$

Max-Independent-Set in Trees

Any tree has at least one leaf (actually two)

For any leaf u in T, there is a max independent set containing u

For any leaf u in T, a max independent set is u union a max independent set in $T \setminus \{u\}$

Algorithm Max Independent set in *Forest F*

$S \leftarrow \emptyset$
while $E(F) \neq \emptyset$ **do**
  Let u be a leaf and v be its neighbor
  $S \leftarrow S \cup \{u\}$
  Remove u, v from $V(F)$ and all edges incident to u and v from $E(F)$

Runtime is $O(n + m)$