Pina Kolling
piko0011

Summary: Lecture 5

Summary for the chapter *8.1.* [6]

Reduction

Examples of NP-problems:

- Travelling Salesman Problem
- SATISFIABLE
- REACHBILITY (in P)
- CIRCUIT VALUE (in P)

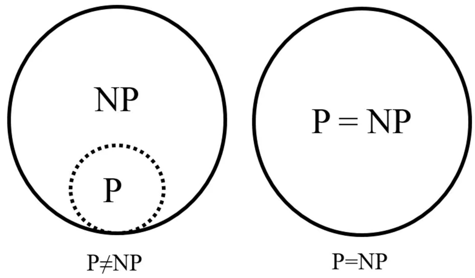


Figure 1: P and NP sets [3]

- reduction: a problem is at least as hard as another
- problem A is at least as hard as problem B if B reduces to A
- B reduces to A if there is a transformation R
  - R produces for every input x of B an equivalent input $R(x)$ of A
  - the answer of input x on B and input $R(x)$ on A have to be the same
- to solve B on input x, A can be solved instead with input $R(x)$

Reduction

Problem A is at least as hard as problem B if B reduces to A.

Transformation function:

- tranformation function R should not be too hard to compute
  $\rightarrow R$ should be limited

- efficient reduction R: $\log n$ space bounded

  Transformation function

  A language L_1 is reducible to L_2 if there is a function R computable by a deterministic Turing Machine in space $O(\log n)$ and $x \in L_1 \Leftrightarrow R(x) \in L_2$.

  R is called a reduction from L_1 to L_2.

- A Turing Machine M that computes a reduction R halts for all inputs x after a polynomial number of steps.
  - there are $O(n \cdot c^{\log n})$ possible configurtions for M on an input of length n
  - deterministic: no configuration can be repeated
  - computation of length at most $O(n^k)$

Pina Kolling
piko0011

Reduction HAMILTONIAN PATH to SATISFIABLE

Problem: HAMILTON PATH

The Hamiltonian Path problem asks whether there is a route in a directed graph G from a start node to an ending node, visiting each node exactly once. (Is there a path in G that visits each node one?) [1]

Problem: SAT

The SAT (satisfiability) problem is the problem of determining if there exists an interpretation that satisfies a given Boolean formula. [7]

- HAMILTON PATH can be reduced to SAT
  $\rightarrow$ demonstrates HAMILTON PATH is not significantly harder that SAT
- construct a boolean expression $R(G)$ that is satisfiable only if G has a Hamilton path
  $\rightarrow$ write a logical formular that only becomes true when HP is true
- instance: Graph $G = (V, E)$ with n nodes $(1, 2, ..., n)$
- $R(G)$ has n^2 boolean variables $x_{i,j}$ then
- node j is the ith node in the HAMILTON PATH
- $R(G)$ is in conjuctive normal form (CNF: $(a \vee b) \wedge (\neg a \vee c)$)
- conjuncted clauses of $R(x)$:
  - each node j must appear in the path $x_{1,j} \vee x_{2,j} \vee ... \vee x_{n,j}$ – for every node j
  - no node j appears twice in the path: $\neg x_{i,j} \vee \neg x_{k,j}$ for all i, j, k with $i \neq k$
  - every position i on the path must be occupied – $x_{i,1} \vee x_{i,2} \vee ... \vee x_{i,n}$ for each i
  - no two nodes j and k occupy the same position in the path – $\neg x_{i,j} \vee \neg x_{i,k}$ for all i, j, k with $j \neq k$
  - nonadjacent nodes i and j cannot be adjacent in the path – $\neg x_{k,i} \vee \neg x_{k+1,j}$ for all $(i, j) \notin E$ and $k = 1, 2, ..., n-1$

  [4]

Proof idea:

- to show:
  - for any graph G, $R(G)$ has a satisfying truth assignment only if and only if G has a Hamilton path
  - R can be computed in space $\log n$
- $R(G)$ contains $O(n^3) clauses$:
  - for each node j exists a unique position i $(x_{i,j})$
  - for each position i exists a unique node j $(x_{i,j})$
  - permutaion π of the nodes with $\pi(i) = j$ if $x_{i,j}$
  - $(\pi(1), \pi(2), ..., \pi(n))$ is a hamilton path
  - the truth assignment $T(x_{i,j}) = true$ if and only if $\pi(i) = j$

Boolean Circuit

A Boolean circuit is a mathematical tree model for logic formulas.
Boolean circuits are defined in terms of the logic gates they contain. For example, a circuit might contain binary AND and OR gates and unary NOT gates, or be entirely described by binary NAND gates.

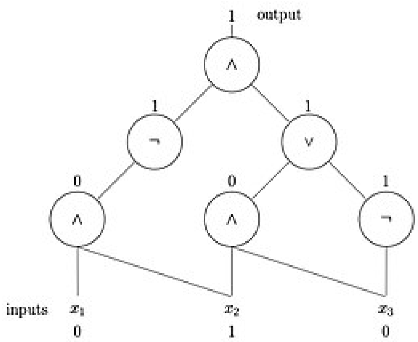


Figure 2: Boolean circuit example [2]

Reduction REACHABILITY to CIRCUIT VALUE

Problem: GRAPH REACHABILITY

Given a graph G and two nodes $n_1, n_2 \in V$, is there path from n_1 to n_2?
A graph $G = (V, E)$ is a finite set V of nodes and a set E of edges as node pairs.

REACHIBILITY can be nondeterministically solved in space $\log n$.

Problem: CIRCUIT VALUE

The CIRCUIT VALUE Problem is the problem of computing the output of a given Boolean circuit on a given input.
In terms of time complexity, it can be solved in linear time (topological sort).
The problem is closely related to the SAT (Boolean Satisfiability) problem which is complete for NP and its complement, which is complete for co-NP.

- REACHABILITY can be reduced to CIRCUIT VALUE
  $\rightarrow$ demonstrates REACHABILITY is not significantly harder that CIRCUIT VALUE
- construct a variable-free circuit $R(G)$ that has *true* as output only if G has a path from the start node to node n
- $R(x)$ uses no $\neg$ gates (monotone circuit)
- (both problema are in P)
- idea: use the Floyd-Warshall algorithm (dynamic programming)
- instance: Graph $G = (V, E)$ with n nodes $(1, 2, ..., n)$
- the gates:

- $g_{i,j,k}$ with $1 \leq i, j \leq n$ and $0 \leq k \leq n$
    - there is a path from node i to node j without passing through a node bigger than k
    - $g_{i,j,0}$ is true if and only if $i = j$ or i and j are neighbours
- $h_{i,j,k}$ with $1 \leq i, j, k \leq n$
    - there is a path from node i to node j passing through k but not any node bigger than k

- $h_{i,j,k}$ is an *and* gate with predecessors $g_{i,k,k-1}$ and $g_{k,j,k-1}$ where $k = 1, 2, ..., n$
- $g_{i,j,k}$ is an *or* gate with predecessors $g_{i,j,k-1}$ and $h_{i,j,k}$ where $k = 1, 2, ..., n$
- $g_{1,n,n}$ is the output gate

  [5]

Proof idea:

- prove by induction on k
  $\rightarrow$ prove that the gates work as described
- claim is true for $k = 0$
- if claim is true for $k - 1$: true for k too

- $g_{1,n,n}$ is true if and only if there is a path from 1 to n and there is a path from 1 to n in G
- R can be computed in $\log n$ space
- circuit $R(G)$ is derived from the Floyd-Warshall algorithm

Reduction CIRCUIT SAT to SAT

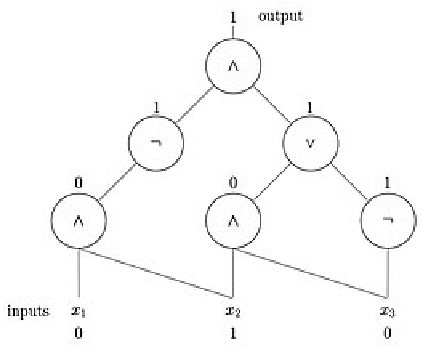
Problem: CIRCUIT SAT
The circuit satisfiability problem (CIRCUIT SAT) is the decision problem of determining whether a given Boolean circuit has an assignment of its inputs that makes the output true.

- CIRCUIT SAT can be reduced to SAT
  $\rightarrow$ demonstrates CIRCUIT SAT is not significantly harder that SAT
- given a circuit C, construct a boolean expression $R(C)$ such that $R(C)$ is satisfiable if and only if C is satisfiable
- the variables of $R(C)$ are those of C plus g for each gate g of C
- clauses of $R(C)$:
    - g is a variable gate x:
      Add clauses $(\neg g \vee x)$ and $(g \vee \neg x) \equiv g \Leftrightarrow x$
    - g is a *true* gate:
      Add clause $(g) \rightarrow g$ must be true to make $R(C)$ true
    - g is a *false* gate:
      Add clause $(\neg g) \rightarrow g$ must be false to make $R(C)$ true
    - g is a $\neg$ gate with predecessor gate h:
      Add clauses $(\neg g \vee \neg h)$ and $(g \vee h) \equiv g \Leftrightarrow \neg h$

- g is a $\vee$ gate with predecessor gates h and h':
  Add clauses $(\neg h \vee g)$, $(\neg h' \vee g)$ and $(h \vee h' \vee \neg g)$, meaning: $g \Leftrightarrow (h \vee h')$
- g is a $\wedge$ gate with predecessor gates h and h':
  Add clauses $(\neg g \vee h)$, $(\neg g \vee h')$, and $(\neg h \vee \neg h' \vee g)$, meaning: $g \Leftrightarrow (h \wedge h')$
- g is the output gate:
  Add clause (g), meaning: g must be true to make $R(C)$ true

[5]

Example:



- g as variable gate: x_1 x_2 x_3
- g as $\wedge$ gate: $(x_1 \wedge x_2)$ $(x_2 \wedge x_3)$
- g as $\neg$ gate: $(\neg x_3)$ $\neg(x_1 \wedge x_2)$
- g as $\vee$ gate: $(x_2 \wedge x_3) \vee \neg x_3$
- g as $\wedge$ gate: $\neg(x_1 \wedge x_2) \wedge ((x_2 \wedge x_3) \vee \neg x_3)$

Figure 3: Boolean circuit example [2]

Generalization

- generalizations are a special form of reductions
- problem A is a generalization of problem B if every instance of B is also an instance of A $\rightarrow B$ can be reduced to A
- inputs of A are a subset of inputs of B and on those inputs A and B have the same answers

Generalization

Problem A is a generalization of problem B if every instance of B is also an instance of A.
This implies B can be reduced to A.
The inputs of A are a subset of inputs of B and on those inputs A and B have the same answers.

Closedness under composition

Closedness under composition

If R is a reduction from a language L_1 to a lanugage L_2 and R' is a reduction from L_2 to a language L_3, then the composition $R \cdot R'$ is a reduction from L_1 to L_3.
$x \in L_1 \equiv R'(R(c)) \in L_3$

Pina Kolling
piko0011

Proof idea:

- $R \cdot R'$ can be computed in space $\log n$
- compose two Turing Machines M and M'
- simulate the machine for R by keeping on a counter which tape position i is
- when M' is about to do the ith write, jump back to the simulation of R and use this value

References

[1] J. Baumgardner, K. Acker, O. Adefuye, and et al. "Solving a Hamiltonian Path Problem with a bacterial computer". In: *J Biol Eng* 3.11 (2009). DOI: `https://doi.org/10.1186/1754-1611-3-11`.

[2] *Image source: Boolean Circuit.* `https://upload.wikimedia.org/wikipedia/en/thumb/d/df/Three_input_Boolean_circuit.jpg/300px-Three_input_Boolean_circuit.jpg`.

[3] *Image source: P-NP sets.* `https://www.techno-science.net/actualite/np-conjecture-000-000-partie-denouee-N21607.html`.

[4] Prof. Yuh-Dauh Lyuu. *Lecture slides on Reduction of hamiltonian path to sat.* `https://www.csie.ntu.edu.tw/~lyuu/complexity/2011/20111018.pdf`. 2011.

[5] Prof. Yuh-Dauh Lyuu. *Lecture slides on Reductions.* `https://www.csie.ntu.edu.tw/~lyuu/complexity/2004/c_20041020.pdf`. 2004.

[6] Christos H. Papadimitriou. *Computational Complexity.* Addison-Wesley Publishing Company, 1994.

[7] Prof. Dr. Thomas Schwentick. *Lecture slides in Grundbegriffe der theoretischen Informatik.* `https://www.cs.tu-dortmund.de/nps/de/Studium/Ordnungen_Handbuecher_Beschluesse/Modulhandbuecher/Archiv/Bachelor_LA_GyGe_Inf_Modellv/_Module/INF-BfP-GTI/index.html`.