Pina Kolling
piko0011

Summary: Lecture 7

Summary for the chapters *9.1* and *9.2*. [3, 1]

NP-Completeness

NP

Class of lanugages decided by nonderteministic Turing machines in polynomial time.
Most problems are in NP.

NP-completeness:

- easiest problems among those we do not know how to solve efficiently
- if P≠NP can be proven: exact border of efficient solvability is found
- best bet for proving P=NP: show that some NP-complete problem is P
- Until then, the NP-complete problems are the least likely ones in NP to be efficiently solved
- Where is the line between P and NP?

Lanugage L

$L = \{x : (x, y) \in R \textit{ for some } y\}$
L gets an input x and finds a y with $((x, y) \in R$ and the relation $R \subseteq \Sigma^* \times \Sigma^*$.

Polynomially decidable:

- R is polynomially decidable if there is da deterministic Turing machine deciding the language L in polynomial time
- then the relation R (not the language L) is polynomially decidable

Polynomially balanced:

- R is polynomially balanced if $(x, y) \in R$ implies $|y| \leq |x|^k$ for some $k \geq 1$
  → length of the second component is bounded by a polynomial in the length of the first
- then the relation R (not the language L) is polynomially balanced

NP

The language $L \subseteq \Sigma^*$ is in NP only if there is a polynomially decidable and polynomially balanced relation R such that $L = \{x : (x, y) \in R \textit{ for some } y\}$.

For example: Is there a satisfying assignment (y) for a formular (x)?

Proof idea:

-

TODO
proof

Succinct certificate (for NP-complete problems)

- *yes* instance of x has a polynomial witness y (certificate)
- *no* instances don't have such a certificate
- Examples:
  - SAT: certificate is the truth assignment
  - HAMILTONPATH: certificate is the hamilton path of a graph

Typical problems in NP

- sometimes the optimum needs to be found
- sometimes any object that fits the specification is enough
- constraints can be added to optimization problems

3Sat is NP-complete

SAT

The SAT (satisfiability) problem is the problem of determining if there exists an interpretation that satisfies a given Boolean formula. [5]

3SAT

Like the SAT problem, 3SAT is determining the satisfiability of a formula in CNF where each clause is limited to at most three literals.

- kSAT with $k \geq 1$ is a special case of SAT

Reduction from SAT to 3SAT: [4]

- the reduction replaces each clause with a set of clauses, each having exactly three literals
- rewrite the clauses of of the input
- example:

$$\begin{aligned} &(x_1) \wedge (x_1 \vee \bar{x_2}) \wedge (x_2 \vee x_3 \vee x_5) \wedge (x_1 \vee x_4 \vee \bar{x_6} \vee \bar{x_7}) \wedge (x_1 \wedge x_2 \wedge \bar{x_3} \vee x_5 \vee x_7) \\ \equiv\ &(x_1 \vee x_1 \vee x_1) \wedge (x_2 \vee x_3 \vee x_5) \end{aligned}$$

3Sat with more retrictions

3SAT remains NP-complete even for expressions in which each variable is restricted to appear at most three times and each literal at most twice.

Proof idea:

- if a variable appears more often than twice:
  introduce new variables and make sure (with the introduction of new clauses) that they have the same truthvalue as the original variable

TODO
bipartie graph or whatever ?

Pina Kolling
piko0011

2Sat in P (graph construction)

2-Sat

Like the SAT and 3-SAT problem, 2-SAT is determining the satisfiability of a formula in CNF where each clause is limited to at most two literals.

- let ψ be an instance of 2SAT (clauses with two literals each)
- construct formular ψ as graph
- the nodes are the variables (node for a and $\neg a$)
- for clauses $(\neg a \vee b) \equiv a \rightarrow b$: edge from the node a to the node b
- paths in G are implications (implication is transitiv)
- ψ is unsatisfiable only if there is a variable x such that there are paths from x to $\neg x$ and from $\neg x$ to x in G

Proof idea:

- the transitivity of the implication is proven
- ψ is unsatisfiable only if there is a variable x such that there are paths from x to $\neg x$ and from $\neg x$ to x in G

- there are paths from x to $\neg x$ and from $\neg x$ to x
- path from x to $\neg x$:
  - transitivity of implication

    

    leads to clause $x \rightarrow \neg x \equiv \neg x \vee \neg x \equiv \neg x$
- path from $\neg x$ to x
  - transitivity of implication

    

    leads to clause $\neg x \rightarrow x \equiv \neg\neg x \vee x \equiv x \vee x \equiv x$
- the two clauses are connected with an logic and (because formula is in CNF) which leads to $\neg x \wedge x$ which is unsatisfiable

- there are no paths from any x to $\neg x$ and back x: no edge is changing from true to false or the other way around
- whenever a node is assigned to a value, all the successors are assigned to the same value and there can be no edge from true to false or from false to true

Pina Kolling
piko0011

2Sat in NL

2Sat is in NL.

Proof idea:

- NL is closed under complement
- show: unsatisfiable expressions can be recognized in NL
- nondeterministically guessing a variable x and check for paths between x and $\neg x$ and back

MaxSat is NP-complete

Max2Sat

Max2Sat is the problem of determining the maximum number of clauses, of a given Boolean formula in CNF with a maximum of 2 variables per clause, that can be made true by an assignment of truth values to the variables of the formula.
Max2Sat is an optimization problem.

Max2Sat is NP-complete.

Example:

$$(x)(y)(z)(w)$$
$$(\neg x \vee \neg y)(\neg y \vee \neg z)(\neg z \vee \neg x)$$
$$(x \vee \neg w)(y \vee \neg w)(z \vee \neg w)$$

- not all clauses can be satisfied

Proof idea:

- any truth assignment that satisfies $x \vee y \vee z$ satisfies 6 or 7 clauses
- the remaining truth assignments $(\neg x \wedge \neg y \wedge \neg z)$ satisfy 4 or 6 clauses
- reduction from 3Sat to MaxSat because $x \vee y \vee z$ needs to be satisfied
- instance ψ of 3Sat
- instance $R(\psi)$ of MaxSat
- group clauses of $R(\psi)$ which are corresponding to a clause of ψ
- set constraint of 7 satisfied clauses

TODO
proof
Questions:

NaeSat is NP-complete

NaeSat

Like SAT, NaeSat consists of a collection of Boolean variables and clauses. NaeSat requires that the values in each clause are not all equal to each other (in other words, at least one is true, and at least one is false). [2]

TODO
Questions:

References

[1] Martin Berglund. *Lecture notes in Computational Complexity.*

[2] Cristopher Moore and Stephan Mertens. *The nature of computation.* OUP Oxford, 2011.

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

[4] Swagato Sanyal. *Reduction from SAT to 3SAT.* `https://cse.iitkgp.ac.in/~palash/2018AlgoDesignAnalysis/SAT-3SAT.pdf`, last opened: 29.11.22.

[5] Prof. Dr. Thomas Schwentick. *Lecture notes 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`.